IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF NEW YORK

INTERNATIONAL BUSINESS MACHINES CORPORATION, *Plaintiff* v. PLATFORM SOLUTIONS, INC., *Defendant*	**Civil Action No. 06 CV 13565 (LAK)** **DECLARATION OF TIBOR L. NAGY IN SUPPORT OF DEFENDANT PSI'S CLAIM CONSTRUCTION BRIEF**

I, Tibor L. Nagy, declare as follows:

I am an associate at Susman Godfrey LLP, and I represent Defendant Platform Solutions, Inc. ("PSI") in this case. I submit this declaration in support of Defendant PSI's Claim Construction Brief.

1. Attached as Exhibit 1 is a one-page chart prepared by PSI that provides a useful overview of the 10 patents-in-suit.

2. Exhibit 2, a true and correct copy of excerpts of the deposition of Hance Huston taken March 13, 2008, is being separately filed under seal as an exhibit to the concurrently-filed Confidential Declaration of Tibor Nagy in Support of PSI's Claim Construction Brief because it contains information designated as "Confidential" pursuant to the Protective Order entered in this case on June 14, 2007.

3. Exhibit 3, a true and correct copy of excerpts of the deposition of Soummya Mallick taken May 7, 2008, is being separately filed under seal as an exhibit to the concurrently-filed Confidential Declaration of Tibor Nagy in Support of PSI's Claim Construction Brief because it contains information designated as "Confidential" pursuant to the Protective Order entered in this case on June 14, 2007.

4. Attached as Exhibit 4 is a true and correct copy of excerpts of IBM's 1999 Claim Construction Brief in *Data General Corp. v. IBM*, 94-cv-12213 (D. Mass.).

5. Attached as Exhibit 5 is a true and correct copy of U.S. Pat. No. 7,117,389, filed

Sept. 18, 2003.

6. Exhibit 6, a true and correct copy of excerpts of the deposition of Ronald Avery taken May 23, 2008, is being separately filed under seal as an exhibit to the concurrently-filed Confidential Declaration of Tibor Nagy in Support of PSI's Claim Construction Brief because it contains information designated as "Confidential" pursuant to the Protective Order entered in this case on June 14, 2007.

7. Attached as Exhibit 7 is a true and correct copy of excerpts of the PowerPC 604 RISC Microprocessor User's Manual.

8. Attached as Exhibit 8 is a true and correct copy of excerpts of an IBM publication titled "Enterprise Systems Architecture/390 Principles of Operation."

9. Attached as Exhibit 9 is a true and correct copy of excerpts of an IBM publication titled "IBM Dictionary of Computing."

10. Exhibit 10, a true and correct copy of IBM's patent license agreement with Intel dated October 1, 1989, is being separately filed under seal as an exhibit to the concurrently-filed Confidential Declaration of Tibor Nagy in Support of PSI's Claim Construction Brief because it contains information designated as "Confidential" pursuant to the Protective Order entered in this case on June 14, 2007.

11. Attached as Exhibit 11 is a true and correct copy of a page from the IBM Terminology web site, letters Q and R, *at* http://www-306.ibm.com/software/globalization/terminology/qr.jsp.

12. Attached as Exhibit 12 is a true and correct copy of excerpts of the *Encyclopedia of Computer Science*.

13. Attached as Exhibit 13 is a true and correct copy of excerpts of The Free On-Line Dictionary of Computing, "register," *at* http://foldoc.org/index.cgi?query=register.

14. Attached as Exhibit 14 is a true and correct copy of excerpts of the *Computer Desktop Encyclopedia,* "PSW," *at* http://www.techweb.com/encyclopedia/defineterm.jhtml?term=PSW.

15. Attached as Exhibit 15 is a true and correct copy of the January 23, 2002 Office

Action from the prosecution history of the '789 Patent.

16. Attached as Exhibit 16 is a true and correct copy of the June 24, 2002 Response to Office Action from the prosecution history of the '789 Patent.

17. Attached as Exhibit 17 is a true and correct copy of the September 26, 2002 Office Action from the prosecution history of the '789 Patent.

18. Attached as Exhibit 18 is a true and correct copy of the February 10, 2003 Response to Office Action from the prosecution history of the '789 Patent.

19. Attached as Exhibit 19 is a true and correct copy of the March 19, 2003 Office Action from the prosecution history of the '789 Patent.

20. Attached as Exhibit 20 is a true and correct copy of the July 10, 2003 Reply to Office Action from the prosecution history of the '789 Patent.

21. Attached as Exhibit 21 is a true and correct copy of the September 29, 2003 Office Action from the prosecution history of the '789 Patent.

22. Attached as Exhibit 22 is a true and correct copy of the March 1, 2004 Preliminary Amendment from the prosecution history of the '789 Patent.

23. Attached as Exhibit 23 is a true and correct copy of the January 6, 2004 Response to Office Action from the prosecution history of the '789 Patent.

24. Attached as Exhibit 24 is a true and correct copy of a page from the "Merriam-Webster Online Dictionary," "routine," *at* http://www.merriam-webster.com/cgi-bin/dictionary?book=Dictionary&va=routine.

25. Attached as Exhibit 25 is a true and correct copy of excerpts of the IEEE Standard Dictionary of Electrical and Electronic Terms.

26. Attached as Exhibit 26 is a true and correct copy of the January 4, 1999 Response from the prosecution history of the '520 Patent.

27. Attached as Exhibit 27 is a true and correct copy of the July 26, 1999 Notice of Allowance from the prosecution history of the '261 Patent.

28. Attached as Exhibit 28 is a true and correct copy of Applicants' Response dated December 11, 1996 from the prosecution history of the '106 Patent.

29. Attached as Exhibit 29 is a true and correct copy of excerpts of the *Comprehensive Dictionary of Electrical Engineering.*

30. Attached as Exhibit 30 is a true and correct copy of excerpts of the *Microsoft Computer Dictionary.*

31. Attached as Exhibit 31 is a true and correct copy of excepts of http://www-306.ibm.com/software/globalization/terminology/tu.jsp#x2042528.

32. Attached as Exhibit 32 is a true and correct copy of E. M. Schwarz, L. Sigal, and T. J. McPherson, "CMOS floating-point unit for the S/390 Parallel Enterprise Server G4," *IBM Journal of Research and Development*, Vol. 41, No. 4/5, pp. 475-488 (1997).

33. Attached as Exhibit 33 is a true and correct copy of Notice of Allowance dated April 11, 1997 from the prosecution history of the '106 Patent.

34. Attached as Exhibit 34 is a true and correct copy of S. Dao-Trong & K. Helwig, "A Single-Chip IBM System/390 Floating-Point Processor In CMOS," *IBM Journal of Research & Development*, Vol. 36, Issue 4, pp. 733-49 (1992).

35. Attached as Exhibit 35 is a true and correct copy of May 29, 2003 Remarks to Amendment from the prosecution history of the '812 Patent.

36. Attached as Exhibit 36 is a true and correct copy of Physical media, layer 1 from the IBM infocenter found at http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp?topic=/com.ibm.znetwork.doc/znetwork_24.html.

37. Attached as Exhibit 37 is a true and correct copy of Hardware connectivity on the mainframe from the IBM infocenter found at http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp?topic=/com.ibm.znetwork.doc/znetwork_57.html.

38. Attached as Exhibit 38 is a true and correct copy of U.S. Pat. No. 5,740,438.

39. Attached as Exhibit 39 is a true and correct copy of U.S. Patent No. 6,040,861.

40. Attached as Exhibit 40 is a true and correct copy of U.S. Patent No. 5,978,916.

41. Attached as Exhibit 41 is a true and correct copy of U.S. Patent No. 6,097,757.

42. Attached as Exhibit 42 is a true and correct copy of an excerpt from the Compact Oxford English Dictionary, "firmware," *at* http://www.askoxford.com/concise_oed/firmware.

43. Attached as Exhibit 43 is a true and correct copy of excerpts of the McGraw-Hill Dictionary of Scientific and Technical Terms.

44. Attached as Exhibit 44 is a true and correct copy of an IBM publication titled "System p5 590 and 595 Technical Overview and Introduction," *available at* http://www.redbooks.ibm.com/redpapers/pdfs/redp4024.

45. Attached as Exhibit 45 is a true and correct copy of an IBM publication titled "L10 Implementation," *available at* http://www.redbooks.ibm.com/redpapers/pdfs/redp4178.

46. Exhibit 46 is being separately filed under seal as an exhibit to the concurrently-filed Confidential Declaration of Tibor Nagy in Support of PSI's Claim Construction Brief because it contains information designated as "Confidential" pursuant to the Protective Order entered in this case on June 14, 2007.

47. Attached hereto as Exhibit 47 is a true and correct copy of a letter from Tibor Nagy to the Court dated May 2, 2008.

48. Attached hereto as Exhibit 48 is a true and correct copy of an IBM article titled "What is interrupt processing?", *available at* http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.zconcepts.doc/zconc_interrupts.html.

49. Attached hereto as Exhibit 49 is a true and correct copy of a page from the IBM Terminology web site, letters E and F, *at* http://www-306.ibm.com/software/globalization/terminology/ef.jsp.

Dated: June 5, 2008

Tibor L. Nagy (TN-9569)

PATENT	TITLE	FILED/ISSUED	OVERVIEW
5,414,851 ('851 patent)	"Method and Means for Sharing I/O Resources by a Plurality of Operating Systems"	Jun. 15, 1992 May 9, 1995	Relates to input/output ("I/O") channels in mainframe computers. Claims a method of increasing the effective number of I/O channels that can be shared by mainframe partitions without intervention from a hypervisor.
5,687,106 ('106 patent)	Implementation of Binary Floating Point Using Hexadecimal Floating Point Unit	Mar. 31 1995 Nov. 11, 1997	Relates to floating point operations in mainframe computers. Claims a "floating point unit" that can support both hexadecimal (IBM) and binary (everyone else) floating point formats.
5,696,709 ('709 patent)	"Program Controlled Rounding Modes"	31 Mar. 1995 9 Dec. 1997	Relates to floating point operations in mainframe computers. Claims an apparatus and method dealing with "rounding modes."
5,825,678 ('678 patent)	"Method and Apparatus for Determining Floating Point Data Class"	31 Mar. 1995 20 Oct. 1998	Relates to floating point operations in mainframe computers. Claims an apparatus and method whereby a processor determines whether a floating point number is or is not in a particular data class(es).
5,953,520 ('520 patent)	"Address Translation Buffer for Data Processing System Emulation Mode"	22 Sept. 1997 14 Sept. 1999	Relates to emulation of one computer instruction set architecture ("ISA") by a computer having another, different ISA. Claims a method and system for the "address translation" that must necessarily occur during emulation.
5,987,495 ('495 patent)	"Method and Apparatus for Fully Restoring a Program Context Following an Interrupt"	7 Nov. 1997 16 Nov. 1999	Relates to an event in a processor known as an "interrupt." Claims an instruction that allows a processor to restore the "program context" from "problem state" rather than "supervisor state" (typically, processors restore their program contexts in supervisor state).
6,009,261 ('261 patent)	"Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor"	Dec. 16, 1997 Dec. 28, 1999	Relates to emulation of one computer instruction set architecture ("ISA") by a computer having another, different ISA. Claims a method and system for "patching" and "preprocessing" sequences of emulated instructions.
6,775,789 B2 ('789 patent)	"Method, System and Program Products for Generating Sequence Values That Are Unique Across Operating System Images"	Jun. 21, 1999 Aug. 10, 2004	Relates to the time-of-day clocks utilized by computer processors. Claims a "Store Clock Extended" instruction for storing a specific format of time-of-day clock values.
6,971,002 ('002 patent)	"Method, System and Product for Booting a Partition Using One of Multiple, Different Firmware Images Without Rebooting Other Partitions"	Aug. 9, 2001 Nov. 29, 2005	Relates to partitioning in mainframe computers. Claims a system and method for storing multiple selectable firmwares which partitions can use to boot.
6,654,812 B2 ('812 patent)	"Communication Between Multiple Partitions Employing Host-Network Interface"	Oct 15, 2001 Nov 25, 2003	Relates to partitioning in mainframe computers. Claims a system and method for routing network packets that are sent from one partition to another without using the network.

UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MASSACHUSETTS

DATA GENERAL CORPORATION,

Plaintiff,

v.

INTERNATIONAL BUSINESS MACHINES CORPORATION,

Defendant.

No. 94-12213-NMG

IBM'S RESPONSE TO DATA GENERAL'S CLAIM CONSTRUCTION MEMORANDUM REGARDING U.S. PATENT Nos. 4,575,797 AND 4,455,603

EVAN R. CHESLER
THOMAS G. RAFFERTY
RICHARD J. STARK
CRAVATH, SWAINE & MOORE
Worldwide Plaza
825 Eighth Avenue
New York, NY 10019
(212) 474-1000

Attorneys for Defendant International Business Machines Corporation

Of Counsel:

Stephen D. Poss, P.C.
(BBO #551760)
GOODWIN, PROCTER & HOAR LLP
Exchange Place
53 State Street
Boston, MA 02109
(617) 570-1886

Morris Waisbrot
William F. Haigney
DAVIS, WEBER & EDWARDS
100 Park Avenue
New York, NY 10017
(212) 685-8000

Christopher A. Hughes
MORGAN & FINNEGAN
345 Park Avenue
New York, NY 10154-0053
(212) 415-8524

Donald J. Rosenberg
Assistant General Counsel
Jennifer Daniels
Senior Counsel
INTERNATIONAL BUSINESS MACHINES CORPORATION
44 South Broadway
White Plains, NY 10601
(914) 288-4035

John O. Mirick
MIRICK, O'CONNELL, DEMALLIE & LOUGEE
1700 Bank of Boston Tower
100 Front Street
Worcester, MA 01608

January 4, 1999

386

instructions. The language of claim 1 directly reflects this aim, claiming "instructions" having specified characteristics, as well as a "processor means" for executing those instructions.

To understand how the language of claim 1 describes a computer system that solves the problems supposedly encountered by conventional computers, some technical context is necessary, much of which has already been provided in Data General's *Markman* brief ("Br.").[18]

The function of a digital data processing system (or "computer system") is to run programs by executing instructions that the system's processor can understand. (Br. at 2-3.) The set of all instructions executable by a processor is referred to as its "machine instruction set" or "machine language". (*Id.* at 4 n.7.) Machine instructions consist of binary code (*i.e.*, strings of 1's and 0's), which is all that a processor can actually understand or execute. (*See id.* at 3.)

In the earliest computers, programmers wrote their programs directly in machine language. (Br. at 9.) Because human beings do not think intuitively in binary form, writing programs in the form of 1's and 0's was extremely time consuming and difficult. (*Id.* at 9.) To overcome this difficulty, programmers developed various high level user programming languages, such as FORTRAN and COBOL. (*Id.*) These languages were "high level" in the sense that they closely resembled human language rather than the stream of binary digits to which a computer processor was able to respond. (*Id.* at 3.)

While high level languages were easier for humans to use, they could not be understood or executed by a computer. Thus, in order to run a high level language program on a computer, the high level language statements first had to be translated (or "compiled") into machine language instructions, which a processor could understand and execute. (*Id.* at 4, 7.) This translation was generally performed by a computer program known as a "compiler". (*Id.* at 9.)

---

[18] To the extent possible, IBM will cite portions of Data General's brief in support of propositions about which there is no material dispute.

Over time, the number of high level language programs available to computer programmers proliferated. Each high level language was imbued with specialized functions that made it particularly suitable for certain uses. For example, the FORTRAN language was developed to handle numerical, scientific computations, while the COBOL language was designed to be used effectively for business applications. The number of high-level languages grew quickly. By 1969, over 100 languages had been developed, with new ones appearing "daily (literally)". Jean E. Sammet, *Programming Languages: History and Fundamentals* vi, 753-64(B715-33) (1969).

Conventional computer systems typically employed a single machine instruction set, in which the meaning of each "operation code" was fixed. An "operation code" is the portion of an instruction that specifies the operation that the processor must perform in carrying out the instruction (such as "add" or "subtract"). Because the meanings of their operation codes were fixed, these conventional instruction sets could not be tailored to a broad range of high level languages without increasing substantially the number and size of the operation codes (and thus the instructions) that would be needed to specify the differing functions of all these various languages.[19] This in turn would create a bottleneck in the time it took the system's processor to access instructions from memory,[20] thereby preventing conventional computers from executing efficiently programs written in a broad range of high level languages.

[19] As mentioned previously, a machine instruction is simply a string of binary digits (or "bits"), represented in the form of 1's and 0's. An operation code, being part of a machine instruction, also consists of 1's and 0's. However, the total number of different operation codes that can be devised from a particular length string of 1's and 0's is limited. For example, if a computer's operation codes may be only 4 bits long, only 16 (*i.e.*, 2^4) different operation codes are possible. Alternatively, if each operation code may be eight bits long, up to 256 (*i.e.*, 2^8) different operation codes can be constructed. If more than 256 were needed, the size of the instruction's operation code field (and thus of the instruction itself) would have to be increased. Hence, to accommodate the differing functions of many different high level languages, both the number and size of a given computer's operation codes would have to be increased.

[20] As Mr. Gruner put it in testimony given before he was designated as Data General's expert, the "fewer" the instructions and the "shorter" the instructions are the less time delay you incur in having to access them from memory. (Gruner Dep. at 40-41(B770-71).)

C. The Required Characteristics of All Instructions in the Patented Computer System

In addition to requiring a memory means that stores and provides items of data, subparagraph (1) of '797 claim 1 reads as follows:

> "said items of data including instructions, each one of said instructions containing an operation code of a plurality of operation codes, said operation codes belong to a plurality of functionally different operation code sets, said operation codes in a given one of said operation code sets being definable solely with reference to said given operation code set, and said instructions having common formats". ('797/4:29-37(A14).)

This language requires that among the items of data stored and provided by the memory means there must be instructions, and every instruction is required to have certain characteristics. Each of the requirements imposed by this claim language is discussed below.

1. "instructions"

The parties dispute the meaning of the word "instructions". IBM contends that the word "instructions" means executable instructions into which programs written in high-level user languages are compiled for execution by the processor. Data General, on the other hand, proposes that the term "S-Instructions" be substituted for the word "instructions". (Br. at 40.)

a. IBM's Contentions

The term "instructions" here refers to the executable machine instructions into which programs written in high level user languages (such as FORTRAN or COBOL) are compiled for execution by the processor. This interpretation is supported by the '797 and '602 specifications and the claim language, and was confirmed by Data General's 30(b)(6) testimony.

The Claim Language. In computer systems, as discussed above (*see supra* at 20), high level language programs are translated (or "compiled") into executable machine instructions, and the processor receives and executes those instructions. Claim 1, on its face, refers to such instructions, since the claim states that instructions are received from the memory by the processor, which then responds to each received instruction by performing the operation

US007117389B2

(12) **United States Patent**
Luick

(10) **Patent No.: US 7,117,389 B2**
(45) **Date of Patent: Oct. 3, 2006**

(54) **MULTIPLE PROCESSOR CORE DEVICE HAVING SHAREABLE FUNCTIONAL UNITS FOR SELF-REPAIRING CAPABILITY**

(75) Inventor: **David Arnold Luick**, Rochester, MN (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 594 days.

(21) Appl. No.: **10/667,084**

(22) Filed: **Sep. 18, 2003**

(65) **Prior Publication Data**

US 2005/0066079 A1 Mar. 24, 2005

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/11**; 714/10

(58) **Field of Classification Search** 711/11, 711/13, 10

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,685,009	A *	11/1997	Blomgren et al.	712/23
5,987,587	A *	11/1999	Meltzer	712/23
6,148,395	A *	11/2000	Dao et al.	712/222
6,691,216	B1 *	2/2004	Kelly et al.	711/167
6,725,354	B1 *	4/2004	Kahle et al.	712/34
6,785,841	B1 *	8/2004	Akrout et al.	714/11

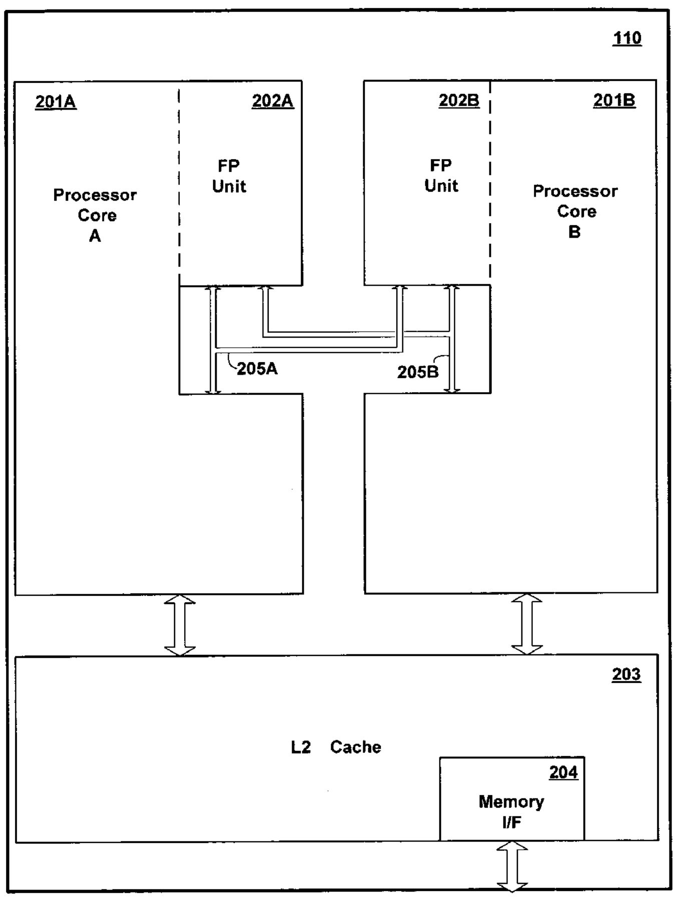
* cited by examiner

Primary Examiner—Bryce P. Bonzo
Assistant Examiner—Marc Duncan
(74) *Attorney, Agent, or Firm*—Roy W. Truelson

(57) **ABSTRACT**

Multiple processor cores are implemented on a single integrated circuit chip, each having its own respective shareable functional units, which are preferably floating point units. A failure of a shareable unit in one processor causes that processor to share the corresponding unit in another processor on the same chip. Preferably, a functional unit is shared on a cycle interleaved basis.

15 Claims, 6 Drawing Sheets



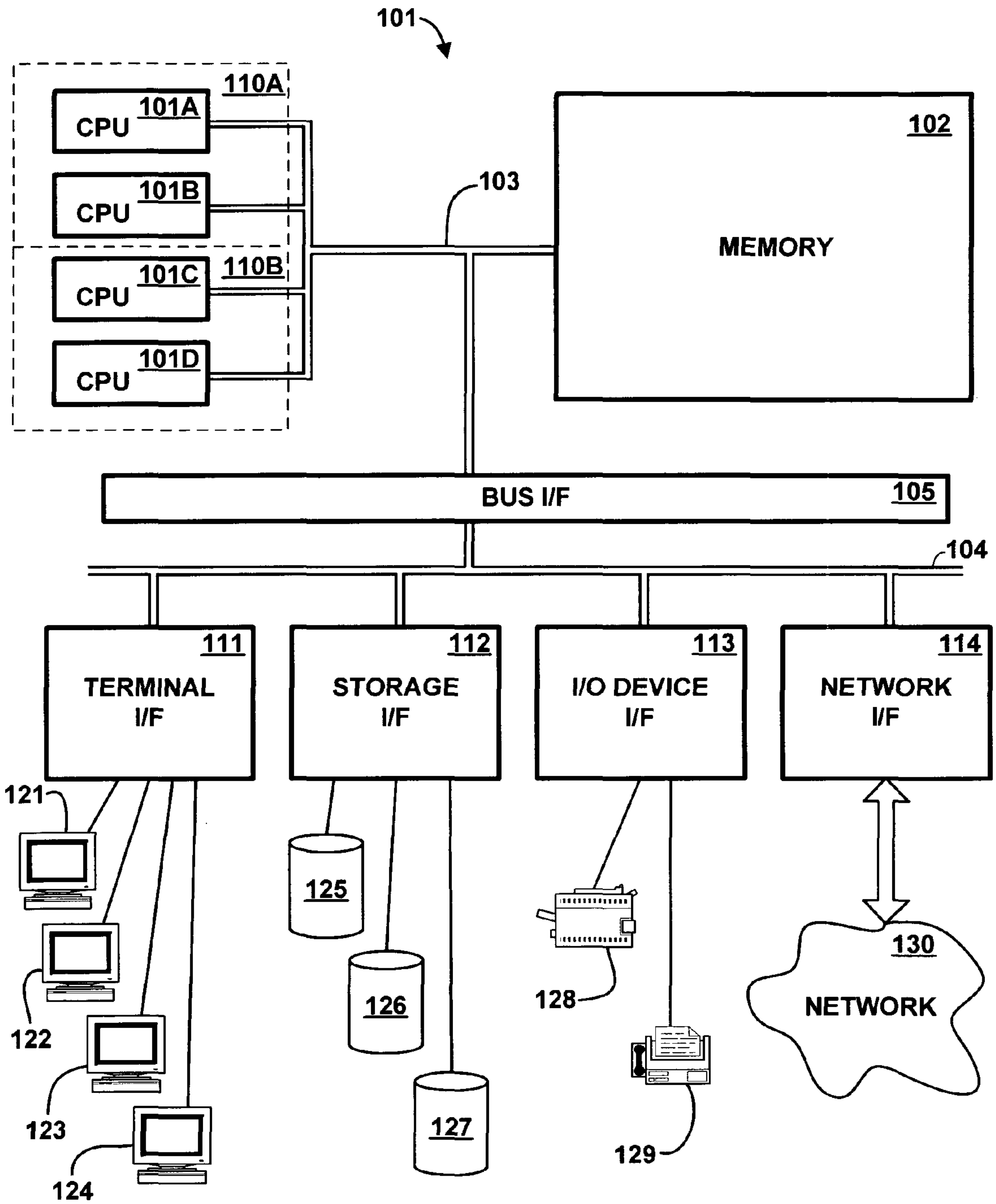

101
110A
CPU 101A
CPU 101B
110B
CPU 101C
CPU 101D
103
MEMORY 102
BUS I/F 105
104
TERMINAL I/F 111
STORAGE I/F 112
I/O DEVICE I/F 113
NETWORK I/F 114
121
122
123
124
125
126
127
128
129
130
NETWORK

FIG.1

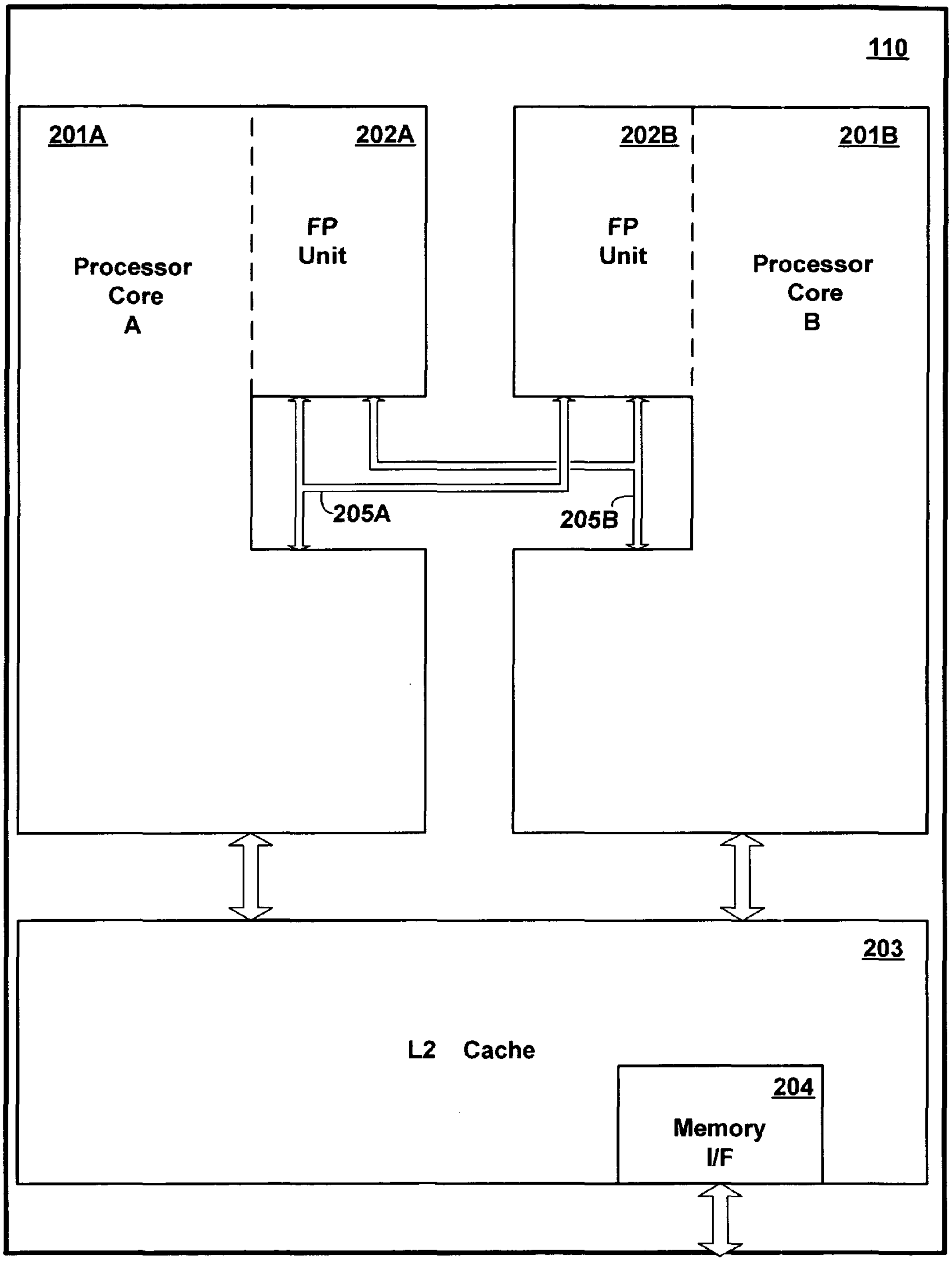

110
201A
202A
FP Unit
Processor Core A
202B
201B
FP Unit
Processor Core B
205A
205B
203
L2 Cache
204
Memory I/F

FIG. 2

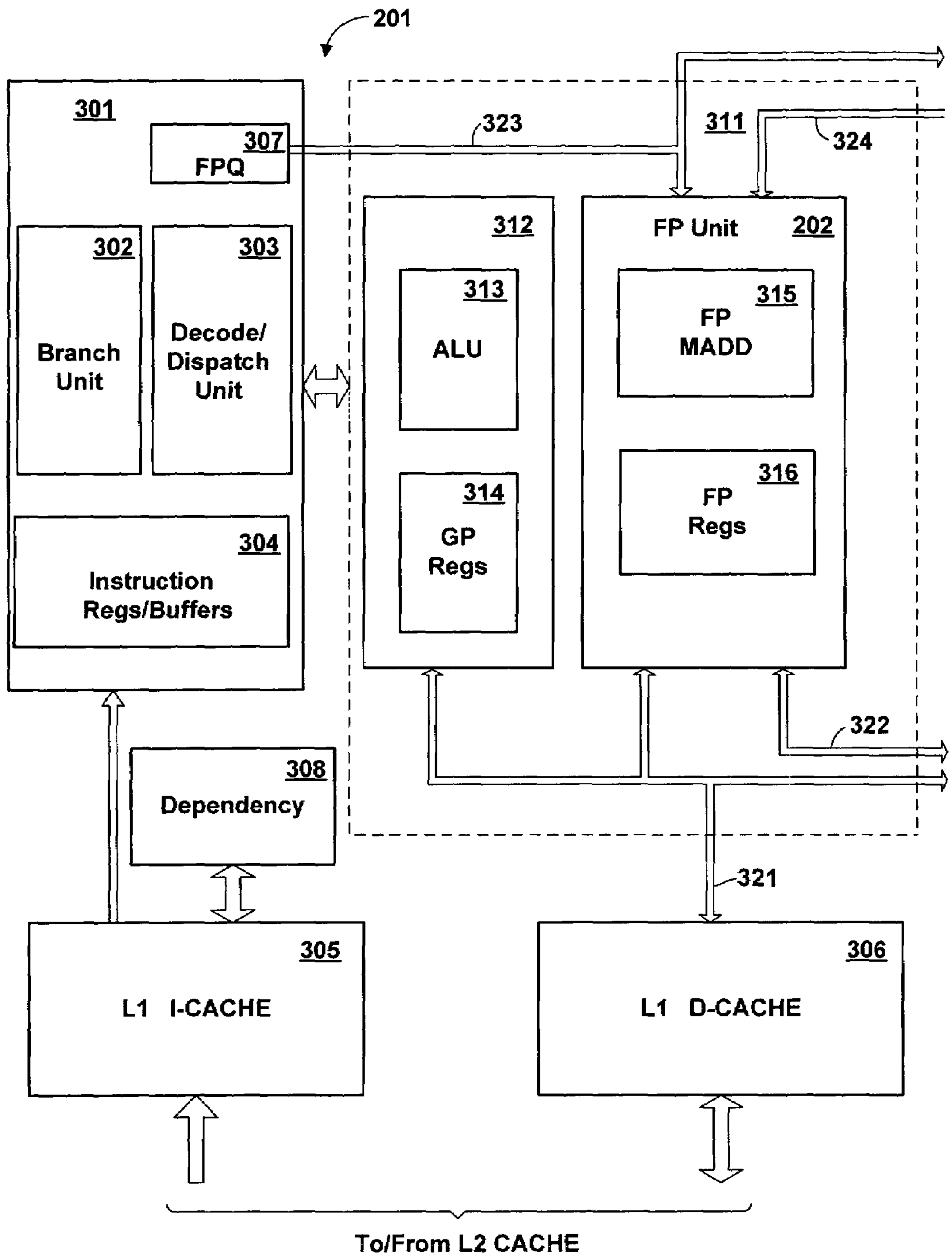
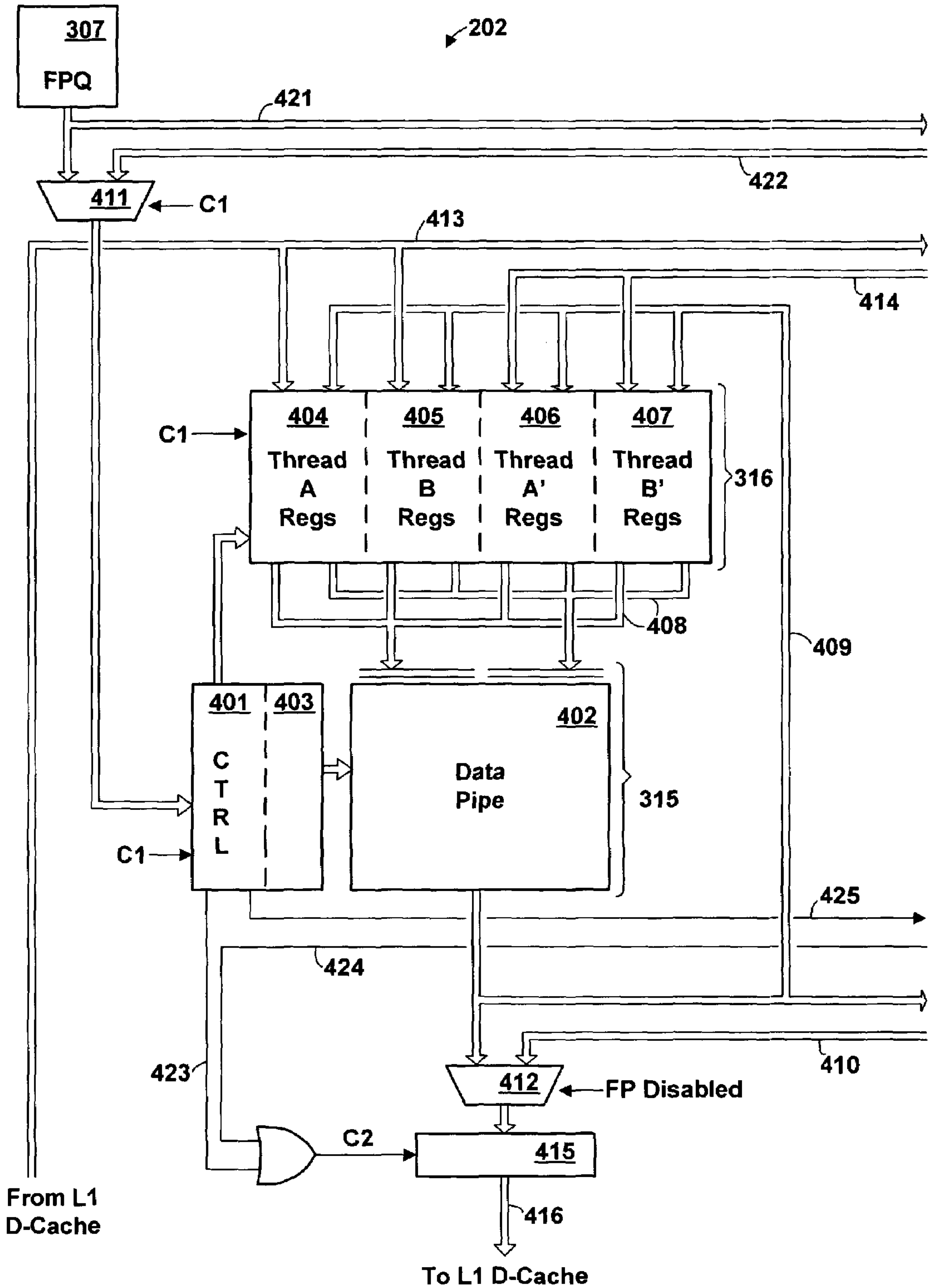

201
301
307
FPQ
323
311
324
302
Branch Unit
303
Decode/ Dispatch Unit
304
Instruction Regs/Buffers
312
313
ALU
314
GP Regs
FP Unit
202
315
FP MADD
316
FP Regs
322
308
Dependency
321
305
L1 I-CACHE
306
L1 D-CACHE
To/From L2 CACHE

FIG. 3



202
307
FPQ
421
422
411
C1
413
414
C1
404
Thread A Regs
405
Thread B Regs
406
Thread A' Regs
407
Thread B' Regs
316
408
409
401
403
C T R L
402
Data Pipe
315
C1
425
424
423
410
412
FP Disabled
C2
415
416
From L1 D-Cache
To L1 D-Cache

FIG. 4

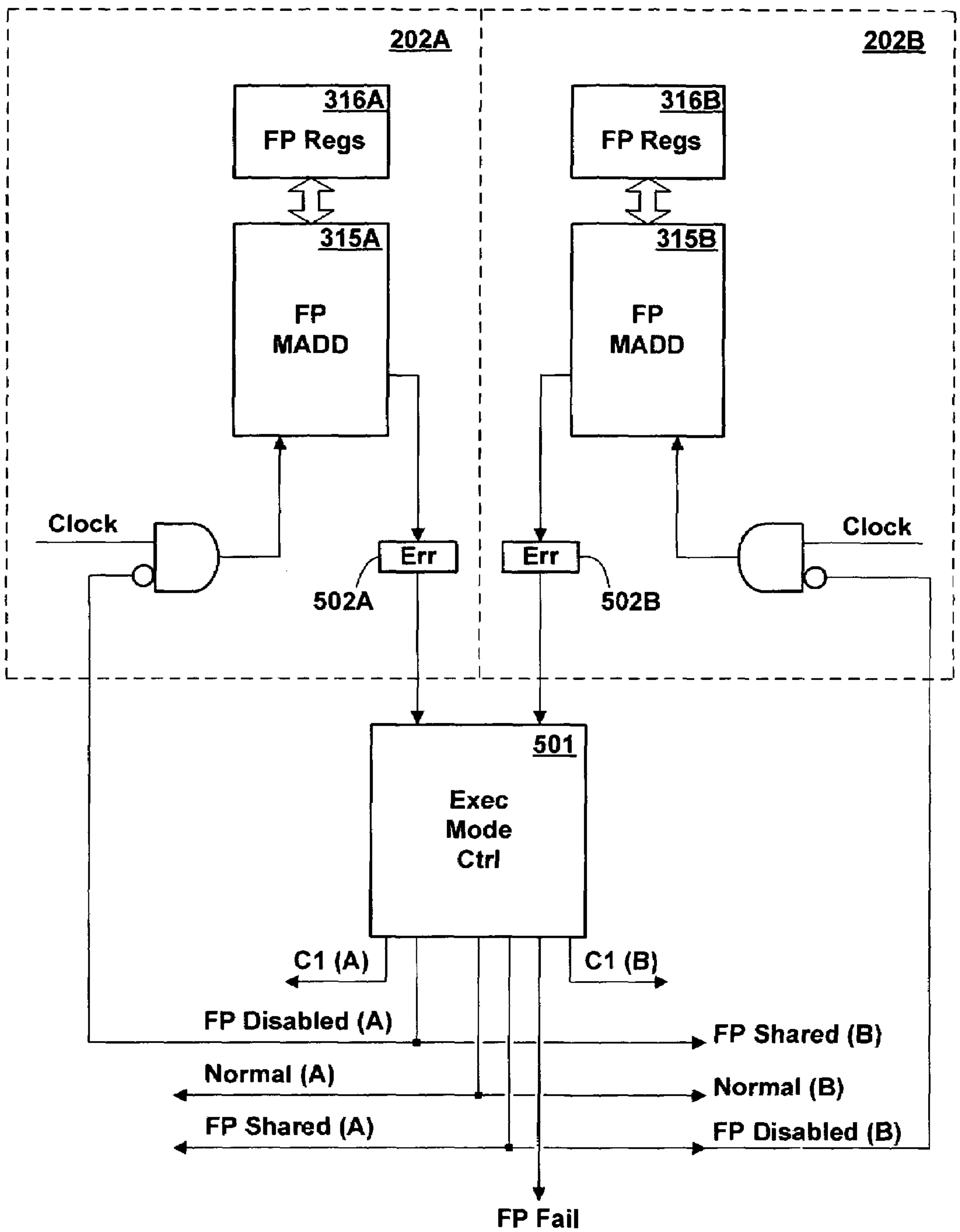
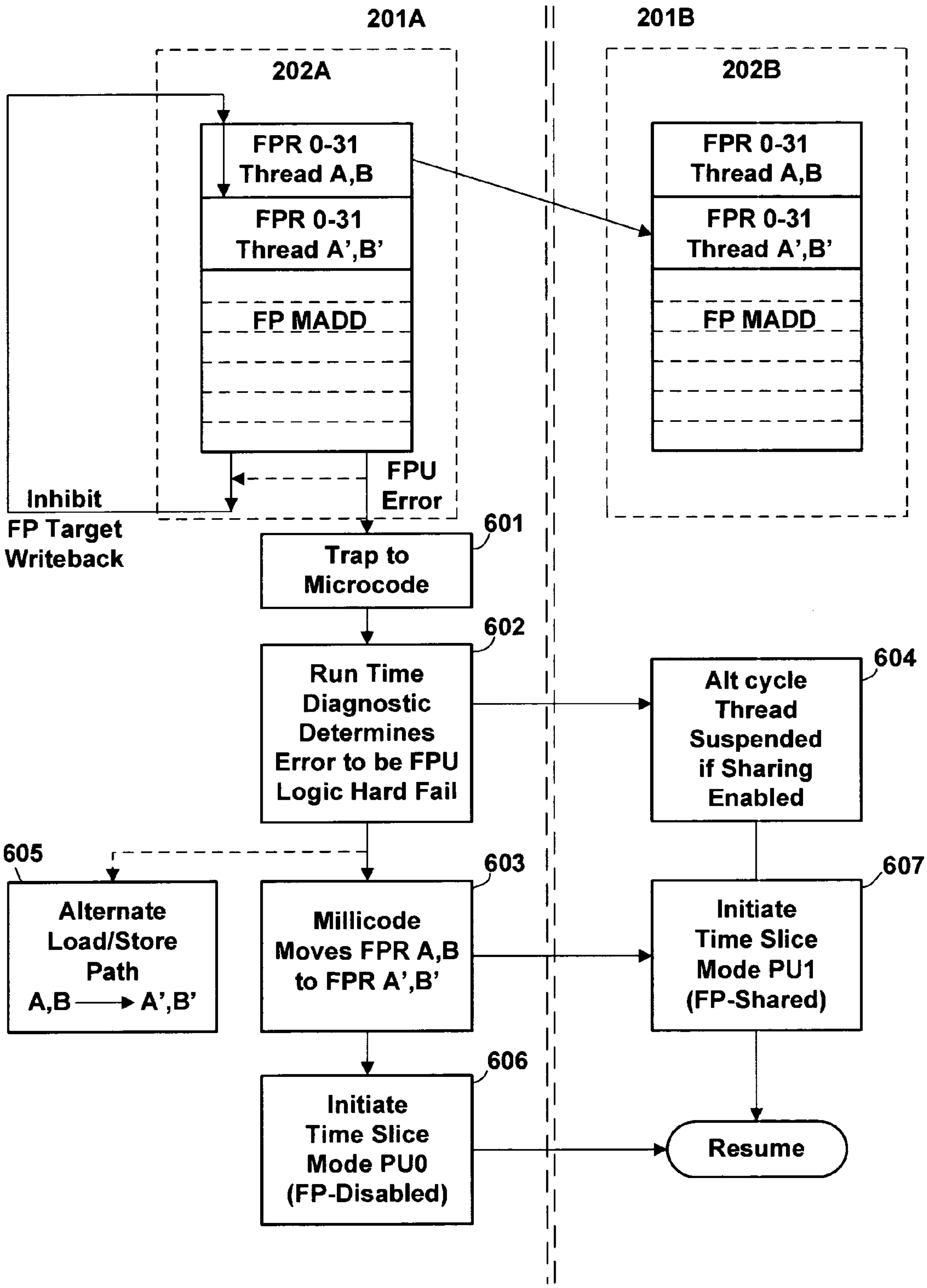

202A
202B
316A
FP Regs
316B
FP Regs
315A
FP
MADD
315B
FP
MADD
Clock
Clock
Err
Err
502A
502B
501
Exec
Mode
Ctrl
C1 (A)
C1 (B)
FP Disabled (A)
FP Shared (B)
Normal (A)
Normal (B)
FP Shared (A)
FP Disabled (B)
FP Fail

FIG.5



201A
201B
202A
202B
FPR 0-31
Thread A,B
FPR 0-31
Thread A',B'
FP MADD
FPR 0-31
Thread A,B
FPR 0-31
Thread A',B'
FP MADD
FPU
Error
Inhibit
FP Target
Writeback
601
Trap to
Microcode
602
Run Time
Diagnostic
Determines
Error to be FPU
Logic Hard Fail
604
Alt cycle
Thread
Suspended
if Sharing
Enabled
605
Alternate
Load/Store
Path
A,B → A',B'
603
Millicode
Moves FPR A,B
to FPR A',B'
607
Initiate
Time Slice
Mode PU1
(FP-Shared)
606
Initiate
Time Slice
Mode PU0
(FP-Disabled)
Resume

FIG. 6

MULTIPLE PROCESSOR CORE DEVICE HAVING SHAREABLE FUNCTIONAL UNITS FOR SELF-REPAIRING CAPABILITY

CROSS-REFERENCE TO RELATED APPLICATION

The present application is related to the following commonly assigned co-pending application, filed on the same date as the present application, which is herein incorporated by reference: Ser. No. 10/667,097 to David A. Luick, entitled "Multiple Parallel Pipeline Processor Having Self-Repairing Capability".

FIELD OF THE INVENTION

The present invention relates to digital data processing hardware, and in particular to the design of multiple processing unit devices for processing data.

BACKGROUND OF THE INVENTION

In the latter half of the twentieth century, there began a phenomenon known as the information revolution. While the information revolution is a historical development broader in scope than any one event or machine, no single device has come to represent the information revolution more than the digital electronic computer. The development of computer systems has surely been a revolution. Each year, computer systems grow faster, store more data, and provide more applications to their users.

A modern computer system typically comprises a central processing unit (CPU) and supporting hardware necessary to store, retrieve and transfer information, such as communications buses and memory. It also includes hardware necessary to communicate with the outside world, such as input/output controllers or storage controllers, and devices attached thereto such as keyboards, monitors, tape drives, disk drives, communication lines coupled to a network, etc. The CPU is the heart of the system. It executes the instructions which comprise a computer program and directs the operation of the other system components.

From the standpoint of the computer's hardware, most systems operate in fundamentally the same manner. Processors are capable of performing a limited set of very simple operations, such as arithmetic, logical comparisons, and movement of data from one location to another. But each operation is performed very quickly. Programs which direct a computer to perform massive numbers of these simple operations give the illusion that the computer is doing something sophisticated. What is perceived by the user as a new or improved capability of a computer system is made possible by performing essentially the same set of very simple operations, but doing it much faster. Therefore continuing improvements to computer systems require that these systems be made ever faster.

The overall speed of a computer system (also called the "throughput") may be crudely measured as the number of operations performed per unit of time. Conceptually, the simplest of all possible improvements to system speed is to increase the clock speeds of the various components, and particularly the clock speed of the processor. E.g., if everything runs twice as fast but otherwise works in exactly the same manner, the system will perform a given task in half the time. Early computer processors, which were constructed from many discrete components, were susceptible to significant speed improvements by shrinking and combining components, eventually packaging the entire processor as an integrated circuit on a single chip. The reduced size made it possible to increase the clock speed of the processor, and accordingly increase system speed.

In addition to increasing clock speeds, it is possible to improve system throughput by using multiple copies of certain components, and in particular, by using multiple CPUs. The modest cost of individual processors packaged on integrated circuit chips has made this practical. While there are certainly potential benefits to using multiple processors, additional architectural issues are introduced. Without delving deeply into these, it can still be observed that there are many reasons to improve the speed of the individual CPU, whether or not a system uses multiple CPUs or a single CPU. If the CPU clock speed is given, it is possible to further increase the speed of the individual CPU, i.e., the number of operations executed per second, by increasing the average number of operations executed per clock cycle.

Most modern processors employ concepts of pipelining and parallelism to increase the clock speed and/or the average number of operations executed per clock cycle. Pipelined instruction execution allows subsequent instructions to begin execution before previously issued instructions have finished, so that execution of an instruction overlaps that of other instructions. Ideally, a new instruction begins with each clock cycle, and subsequently moves through a pipeline stage with each cycle. Because the work of executing a single instruction is broken up into smaller fragments, each executing in a single clock cycle, it may be possible to increase the clock speed. Even though an instruction may take multiple cycles or pipeline stages to complete, if the pipeline is always full, the processor executes one instruction every cycle.

Floating point calculations are among the most time-consuming instructions for processors to perform. In order to increase throughput when handling floating point data, most modern high-performance processors have some form of special floating point functional unit for performing floating point calculations. Due to the complexity of floating point calculations, a floating point unit is generally implemented as a pipeline. Such a pipeline typically requires a significant area of an integrated circuit chip, composed primarily of custom logic. Even though floating point operations are not initiated in every cycle, the circuits within a floating point pipeline have a high degree of switching activity, and consume considerable power at the operating frequencies typical of such devices. The power density, i.e., the amount of power consumed per unit area of chip surface, tends to be significantly greater within the floating point pipelines and similar special units than in many other areas of the processor chip, such as cache arrays and registers. This high level of activity and high power consumption makes the floating point pipeline area of the processor chip particularly susceptible to failure.

In a conventional processor, the failure of any part of a floating point unit generally means that the processor is no longer able to process some subset of the instruction set, i.e., the floating point instructions. Even though only a fraction of the total instructions executed are floating point instructions, if these are interspersed with other instructions in a task thread, the processor will generally be unable to execute it. As a result, the processor may be effectively disabled. This may in turn cause system failure, although in many multiple-processor computer systems, the system can continue to operate, albeit at a reduced throughput, using the remaining functioning processors.

In order to increase the success and acceptability of high-performance processor designs, it is desirable to reduce the frequency of processor failure, and in particular, the frequency of processor failure as a result of failure in some circuitry within a floating point pipeline or similar functional unit. A need exist for improved designs to counter the vulnerability of complex high-performance processors.

SUMMARY OF THE INVENTION

Multiple processor cores are implemented on a single integrated circuit chip, each having its own respective shareable functional units. A failure of a shareable functional unit in one of the processors causes the processor to share the corresponding functional unit in at least one other processor on the same circuit chip.

In one aspect of the preferred embodiment, the shareable functional unit is a floating point pipeline. In general, a new floating point operation is not started in the pipeline with every cycle, since floating point operations constitute only a subset of a larger instruction set. Therefore, a floating point pipeline normally has considerable unused capacity, which is typically more than enough to accommodate the needs of a second processor executing an independent thread of instructions. In the event of failure of a floating point unit in one processor on the circuit chip, a floating point unit in an adjacent processor is shared between the processor to which it is normally assigned and the processor having the failed floating point unit.

In another aspect of the preferred embodiment, a shareable functional unit is shared on a time interleaved basis. Specifically, in the preferred embodiment a new instruction from a first processor may be initiated in the shared functional unit on even clock cycles, and a new instruction from a second processor may be initiated on odd clock cycles. New instructions “may be initiated”, because whether an instruction is actually initiated depends on whether there is a currently pending instruction requiring the shared functional unit in the corresponding processor. If one assumes that, on the average, fewer than half of the instructions in a thread require the services of the shared functional unit, then in general there will be little or no queuing of operations waiting to execute in the shared functional unit, and at most an operation will wait only a single cycle (from even to odd, or vice versa) for an assigned execution time slice.

A set of shareable functional unit processors constructed in accordance with the preferred embodiment of the present invention has the capability to continue operating in the event of failure of a one of the functional units, with relatively little degradation of performance. Moreover, because functional units are shareable, it is not necessary to duplicate a functional unit in order to provide system redundancy. Only a relatively small amount of additional input and output selection circuitry and related control circuitry is required for implementation of such a design.

The details of the present invention, both as to its structure and operation, can best be understood in reference to the accompanying drawings, in which like reference numerals refer to like parts, and in which:

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** is a high-level block diagram of the major hardware components of a computer system utilizing a multiple processor core device having shareable functional units, according to the preferred embodiment of the present invention.

FIG. **2** is a high-level diagram showing the major components of an integrated circuit chip having multiple processor cores, according to the preferred embodiment.

FIG. **3** is a high-level diagram of the major components of a single processor core having a shareable floating point unit, according to the preferred embodiment.

FIG. **4** is a diagram showing in greater detail a shareable floating point unit within one of the processor cores, according to the preferred embodiment.

FIG. **5** illustrates the hardware control logic which controls operating mode for the floating point units, according to the preferred embodiment.

FIG. **6** illustrates the steps taken by a system in response to a hardware error in a floating point unit, according to a preferred system environment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to the Drawing, wherein like numbers denote like parts throughout the several views, FIG. **1** is a high-level representation of the major hardware components of a computer system **100** utilizing a multiple processor core device having shareable functional units, according to the preferred embodiment of the present invention. The major components of computer system **100** include multiple central processing units (CPU) **101**A–**101**D, main memory **102**, terminal interface **111**, storage interface **112**, I/O device interface **113**, and communications/network interfaces **114**, all of which are coupled for inter-component communication via buses **103**, **104** and bus interface **105**.

System **100** contains multiple general-purpose programmable central processing units (CPUs) **101**A–**101**D, herein generically referred to as feature **101**. Physically, processors **101** are implemented as multiple processor core devices **110**A, **110**B, herein generically referred to as feature **110**. I.e., in the preferred embodiment, each multiple processor core device **110** is a monolithic integrated circuit chip containing multiple processors **101**, also referred to as processor cores. In the preferred embodiment, each multiple core device **110** contains two processors **101**. Using current technology, it would be difficult to implement more than two processors on a single chip; however, it will be understood that the present invention is not limited to a chip containing two processors, and that future improvements in integrated circuit technology may make it possible to implement more than two processors on a chip.

Each processor **101** executes instruction stored in memory **102**, and may include one or more levels of cache, some of which may be shared with another processor or processors on the same chip. Memory **102** is a random-access semiconductor memory for storing data and programs. Memory **102** is conceptually a single monolithic entity, it being understood that memory is often a more complex arrangement, such as a hierarchy of caches and other memory devices.

Memory bus **103** provides a data communication path for transferring data among CPUs **101**, main memory **102** and I/O bus interface unit **105**. I/O bus interface **105** is further coupled to system I/O bus **104** for transferring data to and from various I/O units. I/O bus interface **105** communicates with multiple I/O interface units **111**–**114**, which are also known as I/O processors (IOPs) or I/O adapters (IOAs), through system I/O bus **104**. System I/O bus may be, e.g., an industry standard PCI bus, or any other appropriate bus technology. The I/O interface units support communication with a variety of storage and I/O devices. For example,

terminal interface unit **111** supports the attachment of one or more user terminals **121**–**124**. Storage interface unit **112** supports the attachment of one or more direct access storage devices (DASD) **125**–**127** (which are typically rotating magnetic disk drive storage devices, although they could alternatively be other devices, including arrays of disk drives configured to appear as a single large storage device to a host). I/O and other device interface **113** provides an interface to any of various other input/output devices or devices of other types. Two such devices, printer **128** and fax machine **129**, are shown in the exemplary embodiment of FIG. **1**, it being understood that many other such devices may exist, which may be of differing types. Network interface **114** provides one or more communications paths from system **100** to other digital devices and computer systems; such paths may include, e.g., one or more networks **130** such as the Internet, local area networks, or other networks, or may include remote device communication lines, wireless connections, and so forth.

It should be understood that FIG. **1** is intended to depict the representative major components of system **100** at a high level, that individual components may have greater complexity that represented in FIG. **1**, that components other than or in addition to those shown in FIG. **1** may be present, and that the number, type and configuration of such components may vary. Several particular examples of such additional complexity or additional variations are disclosed herein, it being understood that these are by way of example only and are not necessarily the only such variations.

Although main memory **102** is shown in FIG. **1** as a single monolithic entity, memory **102** may in fact be distributed and/or hierarchical, as is known in the art. E.g., memory may exist in multiple levels of caches, and these caches may be further divided by function, so that one cache holds instructions while another holds non-instruction data which is used by the processor or processors. Memory may further be distributed and associated with different CPUs or sets of CPUs, as is known in any of various so-called non-uniform memory access (NUMA) computer architectures. Although memory bus **103** is shown in FIG. **1** as a relatively simple, single bus structure providing a direct communication path among CPUs **101**, main memory **102** and I/O bus interface **105**, in fact memory bus **103** may comprise multiple different buses or communication paths, which may be arranged in any of various forms, such as point-to-point links in hierarchical, star or web configurations, multiple hierarchical buses, parallel and redundant paths, etc. Furthermore, while I/O bus interface **105** and I/O bus **104** are shown as single respective units, system **100** may in fact contain multiple I/O bus interface units **105** and/or multiple I/O buses **104**. While multiple I/O interface units are shown which separate a system I/O bus **104** from various communications paths running to the various I/O devices, it would alternatively be possible to connect some or all of the I/O devices directly to one or more system I/O buses.

Computer system **100** depicted in FIG. **1** has multiple attached terminals **121**–**124**, such as might be typical of a multi-user "mainframe" computer system. Typically, in such a case the actual number of attached devices is greater than those shown in FIG. **1**, although the present invention is not limited to systems of any particular size. Computer system **100** may alternatively be a single-user system, typically containing only a single user display and keyboard input, or might be a server or similar device which has little or no direct user interface, but receives requests from other computer systems (clients).

While various system components have been described and shown at a high level, it should be understood that a typical computer system contains many other components not shown, which are not essential to an understanding of the present invention.

FIG. **2** is a high-level diagram showing the major components of an integrated circuit chip having multiple processor cores, according to the preferred embodiment. Referring to FIG. **2**, integrated circuit chip **110** includes a pair of identical processor cores **201**A, **201**B, herein generically referred to as feature **201**. Each processor core includes a respective shareable functional unit **202**A, **202**B, herein generically referred to as feature **202**. In the preferred embodiment, functional units are floating point units (FP units), i.e., pipelines which perform floating point operations on data. However, a shareable functional unit in accordance with the present invention might be something other than a floating point unit.

Shareable functional units **202**A, **202**B have the capability to substitute for each other in the event that any single one of the shareable units becomes disabled. A data and control path **205**A exists for transferring data and execution control signals between various components of processor core **201**A and its associated floating point unit **202**A. However, data and control path **205**A includes an alternate path to floating point unit **202**B. In the event of a failure of floating point unit **202**A, the alternate path portion of data and control path **205**A can be used to transfer data and execution control signals between processor core **201**A and floating point unit **202**B, allowing floating point unit **202**B to serve both processor cores **201**A, **201**B simultaneously. Similarly, a data and control path **205**B exists for transferring data and execution control signals between various components of processor core **201**B and its associated floating point unit **202**B, and in the event of a failure in floating point unit **202**B, path **205**B can be used to allow processor core **201**B to access floating point unit **202**A.

Chip **110** further includes Level 2 cache (L2 cache) **203**, and memory interface **204**. L2 cache **203** is preferably a non-discriminated cache containing both instructions and non-instruction data. L2 Cache **203** is coupled to memory interface **204**, which loads data from or stores it to an external (i.e., off chip) memory location, which is generally main memory **102**, although it could be another level of cache. L2 cache **203** and memory interface **204** are shared units, to be distinguished from shareable units. I.e., as shown in FIG. **2**, there is one and only one L2 cache **203**, and one and only one memory interface **204**. Both processor cores **201**A, **201**B use the same L2 cache and memory interface. The use of a single memory interface **204** is preferred because it reduces the number of I/O pins required on the chip; the use of a L2 cache (as opposed to two smaller caches) increases the probability of a cache hit. However, it would alternatively be possible for each processor core to have its own L2 cache and external interface.

FIG. **3** is a high-level diagram of the major components of a single processor core **201** within integrated circuit chip **110**, having a shareable floating point unit **202**, according to the preferred embodiment. FIG. **3** shows core **201** in greater detail than is depicted in FIG. **2**. Core **201** includes instruction unit portion **301**, execution unit portion **311**, Level 1 Instruction Cache (L1 I-Cache) **305**, and Level 1 Data Cache (L1 D-Cache) **306**. In general, instruction unit **301** obtains instructions from L1 I-cache **305**, decodes instructions to determine operations to perform, and resolves branch conditions to control program flow. Execution unit **311** performs arithmetic and logical operations on data in registers, and

loads or stores data from L1 D-Cache **306**. L1 I-Cache **305** and L1 D-Cache **306** obtain data from (and, in the case of L1 D-Cache, store data to) shared L2 Cache **203**.

Instruction unit **301** comprises branch unit **302**, instruction decode/dispatch unit **303**, instruction registers and buffers **304**, and floating point instruction queue **307**. Instructions from L1 I-cache **305** are loaded into buffers **304** prior to execution. Depending on the CPU design, there may be multiple buffers (e.g., one for a sequential series of instructions, and others for branch-to locations), each of which may contain multiple instructions. Decode/dispatch unit **303** selects one or more instructions to be dispatched for execution from one of the buffers **304** in a current machine cycle, and decodes the instruction or instructions to determine the operation(s) to be performed or branch conditions. Branch unit **302** controls the program flow by evaluating branch conditions, and refills buffers **304** from L1 I-cache **305**. Floating point queue **307** is a short queue for holding floating point instructions which are awaiting an available cycle for execution in a floating point unit. When a floating point instruction is selected for execution by decode/dispatch unit **303**, it is placed on floating point queue **307**. Floating point queue **307** may contain the full instruction, or it may contain a partially decoded form of the instruction. For example, those bits necessary to identify an instruction as a floating point instruction may be omitted from queue **307**, since all instructions in the queue are necessarily floating point instructions. Preferably, floating point queue **307** also contains a thread ID bit for each instruction, identifying the thread with which the instruction is associated; the thread ID bit is used to identify source or destination registers for the operation.

L1 I-cache **305** and L1 D-cache **306** are separate instruction and data caches providing data to instruction and execution units. Typically, data is taken from or stored to an L1 cache by the instruction or execution unit, and if the data is unavailable in an L1 cache, it is loaded into the L1 cache from shared L2 cache **203**, which in turn obtains it from a location external to chip **110** (e.g., from main memory **102**,). Depending on the processor design, it may be possible to by-pass an L1 cache and load data from L2 cache **203** to an execution or instruction register. Memory interface **204** handles the transfer of data across memory bus **103**, which may be to main memory or to I/O units via bus interface **105**.

Execution unit **311** comprises an integer unit **312** and a floating point unit **202**. Integer unit **312** includes a set of general purpose registers **314** for storing data and an integer arithmetic logic unit (ALU) **313** for performing arithmetic and logical operations on data in GP registers **314**, responsive to instructions decoded by instruction unit **301**. Integer ALU **313** is preferably implemented as multiple (preferably two) pipelines which operate independently, although it might be a single pipeline, or might even be a non-pipelined implementation. Execution unit **311** further includes floating point unit **202** for performing floating point operations. Floating point unit **202** includes a set of floating point registers **316**, and a floating point multiply/add (MADD) pipeline **315**. The operation of floating point unit **202** is described in greater detail herein. In addition to components shown in FIG. **3**, execution unit **311** may include additional special purpose registers and counters, load and store hardware for fetching data from or storing it to cache or memory, control hardware, and so forth. In particular, execution unit **311** may include additional pipelines (not shown) separate from the floating point MADD pipeline **315** and integer ALU pipeline(s) **313**. For example, execution unit **311** may contain one or more load/store pipelines for transferring data between GP registers **314** or floating point registers **315** on the one hand and some form of memory (generally L1 D-Cache **306**) on the other. Additional pipelines, such as an instruction fetch and decode pipeline, may exist within processor core **201**.

Load/store bus **321** provides a communications path for loading data from L1 D-Cache **306** to any of GP Registers **314** or FP Registers **316**, or storing data back to the L1 D-Cache. Load/store bus **321** contains an additional outbound path, whereby in an alternate operating mode data can be loaded from L1 D-Cache **306** to a floating point register (not shown) in the complementary processor core on the same chip **110** as processor core **201**, as explained more fully herein. The complementary processor core similarly contains a load/store bus, a portion of which is shown as feature **322**, which is used to load data from its local L1 D-Cache to its local GP registers and FP registers. In an alternate operating mode, load store bus **322** can also load data from the L1 D-Cache of the complementary processor core to FP registers **316** in processor core **201**.

Floating point instruction bus **323** provides a path for instruction control signals from floating point instruction queue **307** to FP unit **202**, which control the selection of registers and operations performed in FP unit **202**. FP instruction bus **323** contains an additional output path, whereby in an alternate operating mode instructions from floating point queue **307** can be executed in the floating point unit (not shown) of the complementary processor core on the same chip **110** as processor core **201**. The complementary processor core similarly contains a floating point instruction bus, a portion of which is shown as feature **324**, which is used to control the operation of its local floating point unit. In an alternate operating mode, floating point instruction bus **324** can also direct the execution of instructions in floating point unit **202** in processor core **201**.

Data and control path **205**, represented at a high level of abstraction in FIG. **2**, includes load store bus **321** and floating point instruction bus **323**. It may include additional control or status lines (not shown).

While various components of processor core **201** have been described and shown at a high level, it should be understood that the processor core of the preferred embodiment contains many other components not shown, which are not essential to an understanding of the present invention. For example, various additional special purpose registers will be required in a typical design. Furthermore, it will be understood that the processor core of FIG. **3** is simply one example of a CPU architecture, and that many variations could exist in the number, type and arrangement of components within processor core **201**, that components not shown may exist in addition to those depicted, and that not all components depicted might be present in a processor design. For example, the number and configuration of buffers and caches may vary; the number and function of execution unit pipelines may vary; registers may be configured in different arrays and sets; etc.

In the preferred embodiment, processor core **201** is a multithreaded processor supporting the concurrent execution of multiple threads and simultaneous dispatching of instructions from different threads in the same machine cycle. In the preferred embodiment, the concurrent execution of two independent threads is supported, it being understood that this number may vary. Each instruction executed in processor core **201** performs a single primitive operation, such as a load, a store, an integer arithmetic or logical operation using operands from GP registers, a floating point multiply or add using operands from FP registers,

or a branch. Decode/dispatch unit **303** can simultaneously dispatch multiple such instructions in a single machine cycle. In the preferred embodiment, up to four instructions may be dispatched in a single cycle, although this number may vary.

Instructions within each executing thread are grouped in groups of non-dependent sequential instructions. When instructions are loaded into L1 I-Cache **305** from L2 Cache, dependency check unit **308** automatically analyzes the instruction stream to determine which instructions can be executed concurrently or out of sequence. The dependency check unit divides the instruction stream into groups of sequential instructions by placing a stop bit at the last instruction of each group. Within any group, the instructions have no dependencies and can safely be executed out of sequence. Specifically, with respect to each instruction, dependency check unit determines whether there is some dependency upon completion of a previous instruction in the same group. Dependency depends on the type of instruction and the operand(s). Where two different instructions reference the same operand, this often, but not always, creates a dependency. E.g., if both instructions use the same operand as a source for data which is read or input to some pipeline, no dependency exists. But if a first instruction writes to an operand location and a second instruction reads from the same location, the second instruction is clearly dependent on the first, and can not be executed before or simultaneously with the first instruction. If a dependency is found within the same group, the dependency check unit places a stop bit in the immediately preceding instruction, so that the instruction being analyzed will be part of a separate group.

Instruction buffer **304** holds instructions from two independent threads. One thread is designated primary (having a higher priority of execution), and the other secondary. In each machine cycle, decode/dispatch unit may select up to four instructions for dispatch to a corresponding unit (generally a pipeline) which executes the instruction. There are preferably two parallel integer ALU pipelines in integer ALU unit **313**, allowing up to two integer ALU instructions to be dispatched; there is a single floating point pipeline, allowing a single floating point instruction to be dispatched; and so on. Decode/dispatch unit **303** selects instructions from the current group of instruction in the primary thread, to the extent there are unexecuted instructions in the group matching the available pipeline resources. If fewer than four instructions in the primary thread can be matched to the available pipelines, the decode/dispatch unit selects unexecuted instructions from the current group in the secondary thread. When all instructions within a group in either thread have been dispatched for execution, the next group of instructions in that thread is made available for selection on the next machine cycle.

In order to support concurrent execution of multiple threads, a separate set of GP registers **314** and a separate set of FP registers **316** exist for each thread. Additionally, certain other state or special purpose registers (not shown) may be duplicated to support multiple active threads.

In accordance with the preferred embodiment, a pair of processor cores is implemented on the same integrated circuit chip, and each processor core contains a respective shareable floating point unit. In normal operation, the floating point unit within each processor core performs floating point operations responsive to instructions within threads executing in that processor core, i.e., responsive to instructions dispatched by the respective decode/dispatch unit within each processor core. In the event that any single floating point unit becomes inoperable (fails), the complementary floating point unit in the other processor core is placed in a special mode of operation, whereby it is shared by both processor cores. I.e., in this special mode of operation, it executes floating point operations dispatched by both processor cores. Preferably, this is accomplished by interleaving floating point operations from the two cores on a cycle-by-cycle basis, although other forms of sharing would be possible. Thus, each processor core has three modes of operation: a normal mode in which it has exclusive use of its own floating point unit, an FP sharing mode in which it shares its floating point unit with the complementary processor core on the same chip, and an FP disabled mode in which its own floating point unit is disabled, and it shares the floating point unit in the complementary processor core on the same chip.

FIG. **4** is a diagram showing in greater detail some of the major components of a shareable floating point unit **202** within one of processor cores **201**, according to the preferred embodiment. The heart of floating point unit **202** is floating point MADD pipeline **315**, comprising a pipeline control decoder **401**, a data pipe **402**, and destination bits **403**. Data pipe **402** is a multi-stage pipeline for processing floating point data operands responsive to control signals from pipeline control decoder **401**. Control decoder obtains instructions for execution from floating point queue **307**, and decodes the instructions to appropriate control signals for directing data pipe **402**. Destination bits **403** are a pair of bits at each pipe stage, which record the thread ID and processor core (core **201** or its complement) from which an instruction originated. Destination bits follow the data through the various stages of the pipe, hence a separate pair of bits is associated with each stage. When processed data emerges from pipe **402**, destination bits **403** are used to determine the destination for the data. Destination bits **403** are shown as a separate part of control decoder **401** for illustrative purposes, although in fact they would typically be implemented as an integral part of control decoder.

Floating point register file **316** is a bank of registers providing source operand data to, and receiving resultant data from, data pipe **402**. Registers **316** contain four sets of independent registers **404**–**407**, each corresponding to a different thread. Register set **404** corresponds to the primary thread executing in processor core **201**; register **405** corresponds to the secondary thread executing in processor core **201**; register set **406** corresponds to the primary thread executing in the processor core which is the complement of core **201** on the same chip **110**; and register set **407** corresponds to the secondary thread executing in the processor core which is the complement of core **201**. In the preferred embodiment, each FP register set **404**–**407** contains 32 registers capable of being designated as operands, each register containing 64 bits, although these parameters may vary. A floating point instruction decoded for execution by floating point unit **202** may specify any of the 32 registers within a register set as a source or destination operand for a pipeline operation; however, the set itself is determined by the context of the instruction, i.e., the processor core and thread from which the instruction originated. Each instruction may specify two source operands, which are input to data pipe **402** via source operand buses **408** as operands A and B.

Selector logic **411** selects either a floating point operation from floating point queue **307**, or a floating point operation from the corresponding floating point queue in the complementary processor, for execution by floating point unit **202**, responsive to control signal C**1** (explained more fully herein). FP Operation bus **421** transmits an FP operation

from queue **307** to selector **411**, ans also transmits the operation to a corresponding selector in the complementary processor. A similar FP Op bus in the complementary processor (a portion of which is shown as feature **422**) transmits FP operations from the complementary processor's FP queue to selector **411**. The operation selected by selector **411** is input to control decoder **401**, which selects operand registers and initiates operations based on the instruction. Control signal C**1** simultaneously serves as a select input to register file **316**, to select between the local thread sets **404** and **405** on the one hand, or the complementary processor's thread sets **406** and **407**, on the other hand. I.e., an instruction originating in processor **201** is automatically deemed to reference register set **404** (if the thread ID is set to the primary thread) or register set **405** (if the thread ID is set to the secondary thread), while an instruction originating in the complementary processor is deemed to reference either register set **406** or **407**, depending on the thread ID.

Data emerging from data pipe **402** is output via destination bus **409** to register file **316**. The destination bit pair **403** corresponding to the emerging data determines which register set of sets **404**–**407** is the intended destination for the data. Additional paths in destination bus **409** allow data emerging from the pipeline to be directly stored in the L1 D-Cache **306** of processor core **201** via selector **412**, store bus register **415**, and store bus **416**, or in the corresponding L1 D-Cache of the complementary processor. The complementary processor contains a corresponding destination bus, a portion of which is shown as feature **410**. Data bound for the L1 D-Cache passes through selector **412**, which can select either the output of data pipe **402** from bus **409** or the output of the corresponding data pipe in the complementary processor core from bus **410**. When processor core **201** is operating in FP disabled mode, the data pipe output of the complementary processor core is selected; otherwise it selects its own data pipe (data pipe **402**). A selector corresponding to selector **412** exists in the complementary processor. When processor **201** is operating in FP sharing mode (and the complementary processor is necessarily in FP disabled mode), the output of pipe **402** is selected by this complementary selector for writing to the L1 D-Cache of the complementary processor core.

Load bus **413** is used to load data from L1 D-Cache **306** to FP registers **316** responsive to floating point load instructions in FP queue **307**. Load bus **413** further includes a path for loading data to register sets in the complementary processor core. When processor core **201** is operating in FP disabled mode, load bus **413** is thus used to load data from L1 D-Cache to a pair of register sets in the complementary processor. Load bus **413** specifically loads data to register sets **404** and **405**, and does not access register sets **406** and **407**. Register sets **406** and **407** are used for threads executing in the complementary processor when processor **201** is in FP sharing mode (and the complementary processor's floating point unit is disabled). In this case all data for register sets **406** and **407** must be loaded from the L1 D-Cache in the complementary processor. For this purpose, the corresponding load bus in the complementary processor (a portion of which is shown as feature **414**) transfers data from the complementary processor's L1 D-Cache to register sets **406** and **407**.

In the preferred embodiment, when operating in FP sharing mode, floating point operations are taken in alternating machine cycles from the local floating point queue **307** and the corresponding floating point queue in the complementary processor core. Both processor cores on chip **110** are synchronized to a common clock signal. Each processor core is assigned a respective even or odd set of clock cycles. If processor core **201** is arbitrarily assumed to be odd, then control signal C**1** may be expressed as follows:

$$C1 = FP\_\text{Sharing\_Mode AND Even\_Clock\_Cycle}$$

Thus, if processor core **201** is not operating in FP sharing mode, C**1** is always (), meaning that the local FP queue **307** is selected (and that the first half of FP register file, consisting of register sets **404** and **405**, is selected when referencing source operands for source operand bus **408**). If the processor is operating in sharing mode, the local FP queue **307** (along with register sets **404** and **405**) will be selected on odd clock cycles, and the FP queue in the complementary processor (along with register sets **406** and **407**) will be selected on even clock cycles.

Signal C**2** is a clock signal used to clock data into store bus register **415** for writing to L1 D-Cache **306**. In the abstract sense, signal C**2** is the logical AND of the processor clock and the existence of emergent data from a floating point pipe which is intended for processor core **201**. Signal C**2** is produced as the logical OR of two separate signals, one from control unit **401** and the other from the corresponding control unit in the complementary processor core. Control unit **401** produces a signal on line **423** whenever new data emerges from pipe **402** intended for processor core **201**, while the corresponding control unit in the complementary processor core produces such a signal on line **424** whenever data emerges from its respective pipe intended for processor core **201**. Control unit **401** similarly produces a signal on line **425** whenever new data emerges from pipe **402** intended for the complementary processor core (i.e., when operating in FP sharing mode). Thus, in normal operating mode or in FP sharing mode, signal C**2** is a clock signal which rises and falls with the processor cycle clock whenever new data emerges from pipe **402** which is intended for processor core **201**. When operating in FP disabled mode, signal C**2** rises and falls only on those processor clock cycles in which data intended for processor core **201** is emerging from the corresponding pipe in the complementary processor. Since the number of cycles in the data pipe may vary, data does not necessarily always emerge on an odd (or even) cycle. One of the bits in destination bit pair **403** is used to determine the correct processor core to which the emerging data corresponds.

It will be observed that this form of time multiplexing, in which fixed clock cycles are assigned to each processor core, is not necessarily the most efficient. It is possible that floating point operations will back up in one of the queues, while there are no floating point operations to perform in the other queue, and the floating point unit is idled every other cycle. This arrangement is chosen in the preferred embodiment because it considerably simplifies the control logic. The present invention is based on the theory that most transactional processing environments do not have extremely heavy usage of floating point operations, and utilizing the floating point processor at 50% of capacity will not generally cause significant performance problems. A small floating point queue **307** is provided for those cases where several floating point operations execute in close succession. Preferably, this queue contains about 4 instructions, although the number could vary. It would alternatively be possible to use other schemes for sharing a floating point unit. For example, control logic could exist which, when operating in FP sharing mode, chooses a floating point operation from queue **307** on odd clock cycles, and if no

floating point operation is pending in queue **307** on an odd cycle, then chooses an operation from the corresponding floating point queue in the complementary processor core.

FIG. **5** illustrates the hardware control logic which controls operating mode for the floating point units, according to the preferred embodiment. As shown in FIG. **5**, a pair of floating point units are contained on a common integrated circuit chip **110**, each comprising respective floating point registers **316**A, **316**B and floating point MADD units **315**A, **315**B. FP MADD units **315**A, **315**B are coupled to respective error registers **502**A, **502**B, which provide input to execution mode control logic **501**. Additionally, a processor clock signal (not shown) is input to mode control logic. Mode control **501** uses these various inputs to generate mode or control signals for enabling sharing of floating point units.

In the preferred embodiment, each FP MADD unit is associated with a respective error register **502**A, **502**B, containing a hard error bit indicating whether an unacceptable hard error has been detected in the corresponding MADD unit, the unit is considered to be inoperable. Each register **502**A, **502**B may additionally include one or more bits used to record soft errors. In the preferred embodiment, error conditions are detected by checking parity of data quantities at various stages of the pipeline. A single isolated parity error may be considered a “soft” error, but if the parity error is repeated after retrying the operation, or if isolated parity errors appear at an unacceptable frequency, the error condition will be considered a “hard” error, meaning that the floating point unit is inoperable, and the corresponding hard error bit in one of registers **502**A, **502**B will be set.

Additional techniques for detecting error conditions may be used in addition to or in place of parity checking. Various such techniques are known in the art, and an error condition may be detected using any of these various techniques, or any technique hereafter developed. For example, special hardware can detect invalid machine states; selective microcode instructions might exercise unused pipeline portions with test data to verify function; low-level system software may run test data through the pipelines during idle machine cycles; etc.

Execution mode control **501** produces control signals C**1**(A), C**1**(B), Disabled(A), Disabled(B), Normal and FP_Fail. Signal C**1**(A) is the C**1** signal (explained above) used to select a floating point queue input for floating point unit **202**A, while signal C**1**(B) is the C**1** signal for floating point unit **202**B. Signal DisabledA indicates that processor core **202**A is operating in FP disabled mode and processor core **202**B is in FP sharing mode; this signal is generated by control **501** if register **502**A indicates a hard error and register **502**B does not. Similarly, signal DisabledB indicates that processor core **202**B is operating in FP disabled mode and processor core **202**A is in FP sharing mode, and is generated if register **502**B indicates a hard error and register **502**A does not. Signal Normal indicates both processors are in normal mode, i.e., neither register **502**A, **502**B indicates a hard error.

Signal FP_Fail indicates a hard error has been detected in both floating point units and is recorded in both registers **502**A, **502**B, i.e., neither FP unit is functioning properly. Depending on the system environment, an FP_Fail signal may cause both processor cores to become inoperable. However, in a preferred system environment, it is possible to assign all execution threads containing any floating point instructions to different processors, so that the processor pair which caused the FP_Fail signal to be generated continue to operate, but process only execution threads without any floating point instructions.

The clock input signals to the FP MADD units are gated by the complement of the respective FP disable signals. If operating in FP_Disable mode, the clock to the FP MADD unit is disabled, so that no activity takes place in the FP MADD unit, thus reducing power consumption.

In operation, the floating point units are continually or periodically monitored using any technique as described, and a hard error in one of the units is recorded in the corresponding error detection register **502**A, **502**B. Assuming only one unit of the pair is non-functioning, data is transferred from the registers in the non-functioning unit to the good unit, and FP sharing is enabled in the good unit. These operations may be accomplished entirely in hardware, without intervention by the operating system.

FIG. **6** illustrates the actions taken upon detection of a hard error. Unit **202** A is arbitrarily chosen as the unit experiencing an error in the example of FIG. **6**, it being understood that the mirror image operations would be performed in the case of an error in unit **202**B. As shown in FIG. **6**, an error detected in unit **202**A causes a trap to microcode (block **601**), and a simultaneous inhibit of the write back of the output of the FP MADD unit (since the output can not be considered reliable). The trap to microcode triggers a set of run-time diagnostics to determine whether the error is in fact a hard error (unrecoverable) in the FP unit (block **602**). If a hard error is confirmed, the contents of the FP register sets for threads A and B within floating point unit **202**A are transferred to the alternate register sets (threads A' and B') within complementary floating point unit **202**B. At the same time, floating point unit **202**B suspends dispatching of new threads in the odd clock cycles to clear the pipe, so that new floating point operations are dispatched from the FP queue in processor **201**A only on the even cycles (block **604**). The alternate load/store paths from the L1 D-Cache in processor **201**A to FP unit **202**B are enabled (block **605**). The time sliced mode of operation is then enabled in both processors (blocks **606**,**607**). Processor **201**A operates in FP disabled mode, whereby operations in its FP queue and data from its L1 D-Cache are transferred to FP unit **202**A for processing, the operations being started in the pipe every odd cycle. Processor **202**B operates in FP sharing mode, whereby it operates normally as before, except that it starts a new operation from its own queue in its own pipe every even cycle.

As described in the preferred embodiment above, an alternate pair of register sets **406**, **407** is required in each of the FP units to hold data from the complementary processor in the event that the FP unit needs to operate in FP sharing mode. This arrangement provides an effective means for sharing pipeline function, but it will be recognized that it requires a doubling of the FP registers, where the alternate register sets are not normally used. Certain variations are possible which would avoid a doubling of the FP registers, but each of these has its own attendant disadvantages. For example, it would alternatively be possible to use only a pair of register sets for threads A and B, and in the event of sharing, assign the thread B register set to the complementary processor’s primary thread data. In this case, only the primary thread in each processor could execute floating point operations. Ideally, threads would be assigned so that only threads without any floating point operations would be assigned as secondary threads. However, if the system architecture does not support such a capability, then a further variation of this alternative would be possible. If a floating

point operation is encountered in a secondary thread and there are no pending floating point operations for the primary thread in the queue, a context switch could be performed in the floating point registers. I.e., the floating point register set corresponding to the processor in which a secondary thread has a floating point operation to perform could be swapped out to L1 D-Cache, and the secondary thread's data loaded into those registers. This obviously would involve some performance penalty, but it would allow the system to continue processing.

In the preferred embodiment described above, a shareable functional unit of a processor core is a floating point processing unit. However, it will be understood that a shareable functional unit in accordance with the present invention could be something other than a floating point unit. For example, a shareable unit might be an integer processing unit, or might be a combined vector processing unit which performs both integer and floating point arithmetic, or might be some other unit.

In the preferred embodiment described above, each processor core **201** supports the concurrent execution of multiple (preferably two) threads, and can dispatch multiple primitive instructions from either or both threads in a single machine cycle. However, it will be appreciated that these are simply processor design implementation details associated with a single embodiment, and that many variations may exist in processor design. By way of illustration, several such variations are mentioned herein, these being intended merely as examples and not as an exhaustive list of processor design variations. A processor according to the present invention might support more than two threads, or might support only a single thread of execution. Moreover, such a processor might be a design in which only one instruction is dispatched for execution in any one machine cycle. Such a processor might use any of various different multithreading techniques, such as fine-grained multithreading in which threads are rotated (usually on a cycle-by-cycle basis), or coarse-grained multithreading in which thread switches are triggered upon the occurrence of latency or other pre-defined events. Additionally, such a processor might be designed having a complex instruction format in which multiple operations are specified in each instruction, as in any of various Single Instruction—Multiple Data (SIMD), Multiple Instruction—Multiple Data (MIMD), Very Long Instruction Word (VLIW) and/or Wide Issue Superscalar processor designs.

Although a specific embodiment of the invention has been disclosed along with certain alternatives, it will be recognized by those skilled in the art that additional variations in form and detail may be made within the scope of the following claims:

What is claimed is:

1. A processing device, comprising:

a plurality of processor cores on a common integrated circuit chip, each processor core containing a respective shareable functional unit for performing operations of a first type; and

control logic detecting a failure of a first of said shareable functional units in a first processor core of said plurality of processor cores, and responsive to detecting said failure, placing a second functional unit in second processor core of said plurality of processor cores in a shared mode of operation, wherein said second functional unit executes operations of said first type originating in said first processor core concurrently with executing operations of said first type originating in said second processor core.

2. The processing device of claim **1**, wherein said shareable function units are floating point units for performing floating point operations.

3. The processing device of claim **1**, wherein said second function unit comprises:

a set of primary input registers for holding input data to said second functional unit from said second processor core; and

a set of alternate input registers for holding input data to said second functional unit from said first processor core when said second functional unit is in said shared mode of operation.

4. The processor of claim **1**, wherein said processing device contains two processor cores.

5. The processing device of claim **1**, wherein said shared mode of operation comprises sharing said second functional unit of a cycle-interleaved basis.

6. The processing device of claim **5**, wherein said shareable functional units are pipelines, and wherein said shared mode of operation comprises initiating an operation in said second functional unit from different respective processor cores on a cycle-interleaved basis.

7. The processing device of claim **1**, wherein said control logic further detects a failure of said second shareable functional unit, and responsive to detecting said failure of said second shareable functional unit, places said first functional unit in a shared mode of operation, wherein said first functional unit executes operations of said first type originating in said second processor core concurrently with executing operations of said first type originating in said first processor core.

8. The processing device of claim **7**, wherein said first function unit comprises:

a set of primary input registers for holding input data to said first functional unit from said first processor core; and

a set of alternate input registers for holding input data to said first functional unit from said second processor core when said first functional unit is in said shared mode of operation.

9. The processing device of claim **1**, wherein said first and second processing cores comprise respective functional unit operation queues, each queue holding a plurality of operations for execution by a shareable functional unit.

10. The processing device of claim **1**, wherein each said processor core supports the concurrent execution of a plurality of threads.

11. The processing device of claim **1**, wherein, if said control logic detects a failure in neither said first functional unit nor said second functional unit, both said first functional unit and said second functional unit operate in a normal mode of operation, wherein said first functional unit executes operations of said first type originating only in said first processor core, and said second functional unit executes operations of said first type originating only in said second processor core.

12. A digital data processing system, comprising:

a memory;

at least one multiple-processor-core device, each said-multiple-processor core device comprising:

(a) a plurality of processor cores on a common integrated circuit chip, each processor core executing at least one respective thread of instructions stored in

said memory, each processor core containing a respective shareable functional unit for performing operations of a first type; and

(b) control logic detecting a failure of a first of said shareable functional units in a first processor core of said plurality of processor cores, and responsive to detecting said failure, placing a second functional unit in a second processor core of said plurality of processor cores in a shared mode of operation, wherein said second functional unit executes operations of said first type originating in said first processor core concurrently with executing operations of said first type originating in said second processor core; and

at least one communications bus transmitting data between said memory and said at least one multiple-processor-core device.

13. The digital data processing system of claim **12**, wherein said digital data processing system comprises a plurality of said multiple-processor-core devices.

14. The digital data processing system of claim **12**, wherein said shared mode of operation comprises sharing said second functional unit of a cycle-interleaved basis.

15. The digital data processing system of claim **12**, wherein said control logic further detects a failure of said second shareable functional unit, and responsive to detecting said failure of said second shareable functional unit, places said first functional unit in a shared mode of operation, wherein said first functional unit executes operations of said first type originating in said second processor core concurrently with executing operations of said first type originating in said first processor core.

* * * * *

MPR604UMU-01

MPC604UM/AD

PowerPC™ 604

RISC Microprocessor User's Manual

IBM Microelectronics PowerPC MOTOROLA

This document contains information on a new product under development. Motorola reserves the right to change or discontinue this product without notice. Information in this document is provided solely to enable system and software implementers to use PowerPC microprocessors. There are no express or implied copyright licenses granted hereunder to design or fabricate PowerPC integrated circuits or integrated circuits based on the information in this document.

The PowerPC 604 microprocessor embodies the intellectual property of IBM and of Motorola. However, neither party assumes any responsibility or liability as to any aspects of the performance, operation, or other attributes of the microprocessor as marketed by the other party. Neither party is to be considered an agent or representative of the other party, and neither has granted any right or authority to the other to assume or create any express or implied obligations on its behalf. Information such as data sheets, as well as sales terms and conditions such as prices, schedules, and support, for the microprocessor may vary as between IBM and Motorola. Accordingly, customers wishing to learn more information about the products as marketed by a given party should contact that party.

Both IBM and Motorola reserve the right to modify this manual and/or any of the products as described herein without further notice. Nothing in this manual, nor in any of the errata sheets, data sheets, and other supporting documentation, shall be interpreted as conveying an express or implied warranty, representation, or guarantee regarding the suitability of the products for any particular purpose. The parties do not assume any liability or obligation for damages of any kind arising out of the application or use of these materials. Any warranty or other obligations as to the products described herein shall be undertaken solely by the marketing party to the customer, under a separate sale agreement between the marketing party and the customer. In the absence of such an agreement, no liability is assumed by the marketing party for any damages, actual or otherwise.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals," must be validated for each customer application by customer's technical experts. Neither IBM nor Motorola convey any license under their respective intellectual property rights nor the rights of others. The products described in this manual are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the product could create a situation where personal injury or death may occur. Should customer purchase or use the products for any such unintended or unauthorized application, customer shall indemnify and hold IBM and Motorola and their respective officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola or IBM was negligent regarding the design or manufacture of the part.

Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

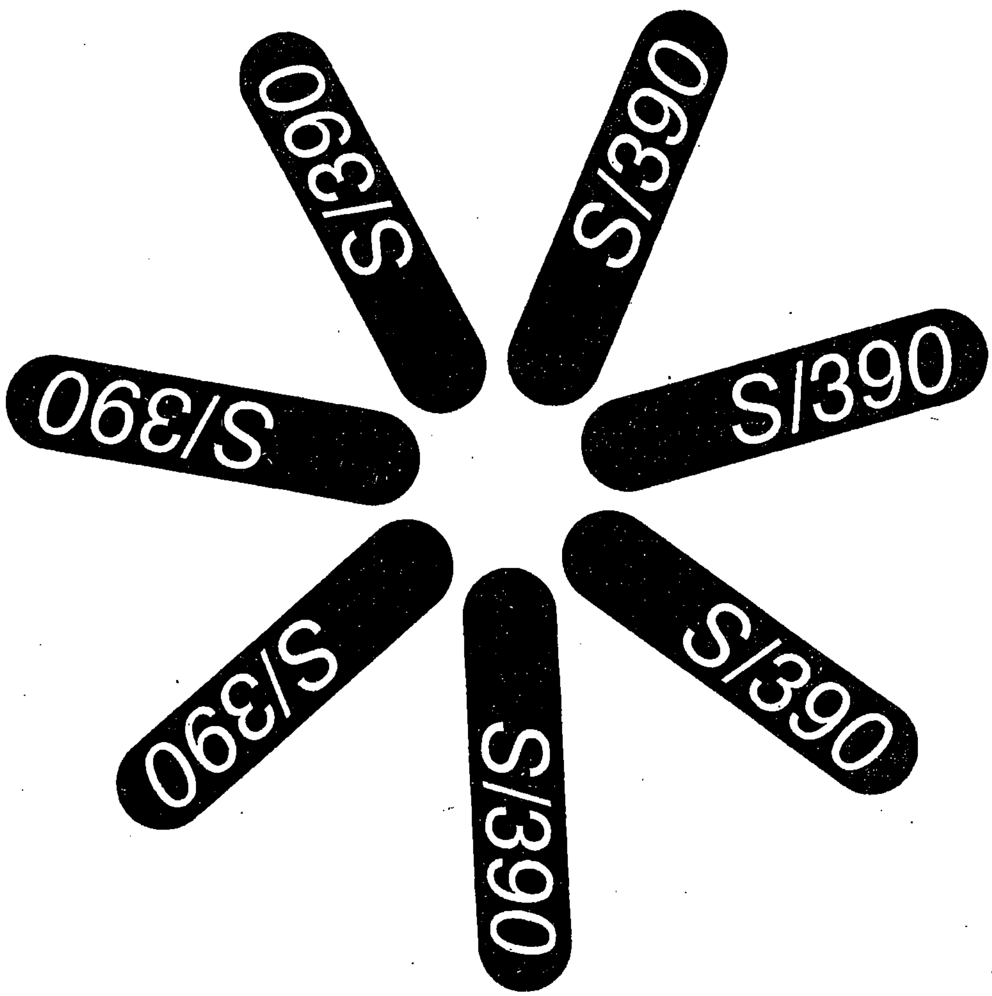
IBM is a registered trademark, and IBM Microelectronics is a trademark of IBM Corp.

PowerPC, PowerPC Architecture, POWER Architecture, PowerPC 601, PowerPC 603, PowerPC 604, and **PowerPC** are trademarks of IBM Corp. used by Motorola under license from IBM Corp.

Enterprise Systems Architecture/390

SA22-7201-02

Principles of Operation



Note:

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xv.

Third Edition (December 1994)

This edition obsoletes and replaces *Enterprise Systems Architecture/390 Principles of Operation*, SA22-7201-01.

This publication is provided for use in conjunction with other relevant IBM publications, and IBM makes no warranty, express or implied, about its completeness or accuracy. The information in this publication is current as of its publication date but is subject to change without notice.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to:

IBM Corporation
Enterprise Systems Central Architecture
Department E57
522 South Road
POUGHKEEPSIE NY 12601-5400
U.S.A.

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Chapter 2. Organization

Main Storage 2-2
Expanded Storage 2-2
CPU 2-2
PSW 2-3
General Registers 2-3
Floating-Point Registers 2-3
Control Registers 2-4
Access Registers 2-4
Vector Facility 2-6
Cryptographic Facility 2-6
External Time Reference 2-6
I/O 2-6
Channel Subsystem 2-6
Channel Paths 2-6
I/O Devices and Control Units 2-7
Operator Facilities 2-7

Logically, a system consists of main storage, one or more central processing units (CPUs), operator facilities, a channel subsystem, and I/O devices. I/O devices are attached to the channel subsystem through control units. The connection between the channel subsystem and a control unit is called a channel path.

A channel path employs either a parallel-transmission protocol or a serial-transmission protocol and, accordingly, is called either a parallel or a serial channel path. A serial channel path may connect to a control unit through a dynamic switch that is capable of providing different internal connections between the ports of the switch.

Expanded storage may also be available in the system, a vector or cryptographic unit may be included in a CPU, and an external time reference (ETR) may be connected to the system.

The physical identity of the above functions may vary among implementations, called "models." Figure 2-1 depicts the logical structure of a two-CPU multiprocessing system that includes expanded storage, a vector unit, and a cryptographic unit and that is connected to an ETR.

Specific processors may differ in their internal characteristics, the installed facilities, the number of subchannels, channel paths, and control units which can be attached to the channel subsystem, the size of main and expanded storage, and the representation of the operator facilities.

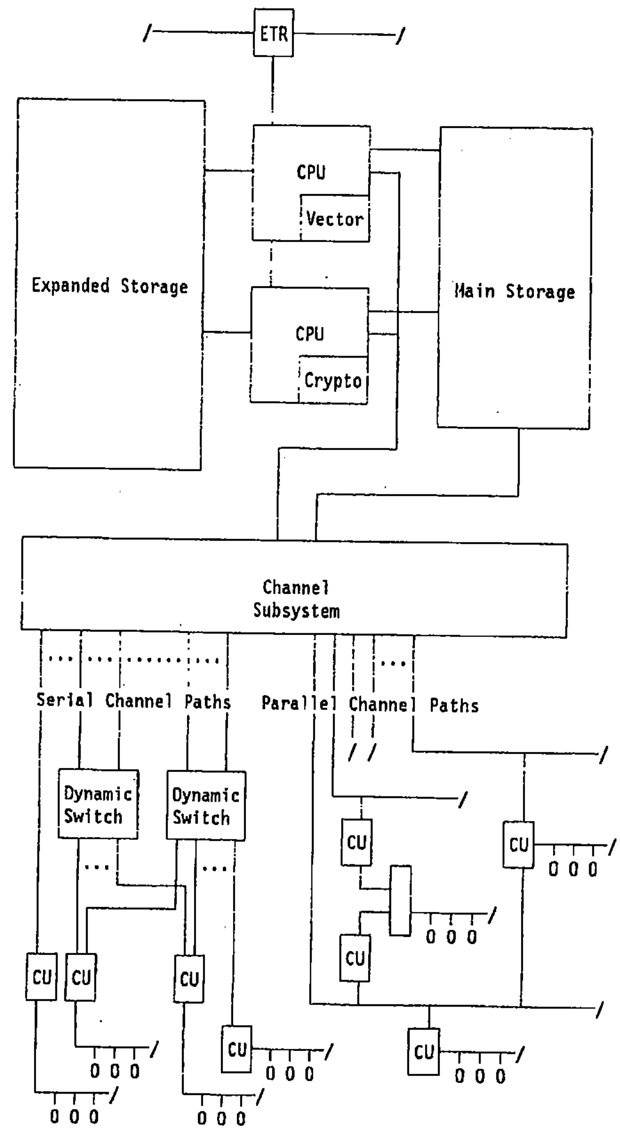


Figure 2-1. Logical Structure of an ESA/390 System with Two CPUs

A system viewed without regard to its I/O devices is referred to as a configuration. All of

the physical equipment, whether in the configuration or not, is referred to as the installation.

Model-dependent reconfiguration controls may be provided to change the amount of main and expanded storage and the number of CPUs and channel paths in the configuration. In some instances, the reconfiguration controls may be used to partition a single configuration into multiple configurations. Each of the configurations so reconfigured has the same structure, that is, main and expanded storage, one or more CPUs, and one or more subchannels and channel paths in the channel subsystem.

Each configuration is isolated in that the main and expanded storage in one configuration is not directly addressable by the CPUs and the channel subsystem of another configuration. It is, however, possible for one configuration to communicate with another by means of shared I/O devices or a channel-to-channel adapter. At any one time, the storage, CPUs, subchannels, and channel paths connected together in a system are referred to as being in the configuration. Each CPU, subchannel, channel path, main-storage location, and expanded-storage location can be in only one configuration at a time.

Main Storage

Main storage, which is directly addressable, provides for high-speed processing of data by the CPUs and the channel subsystem. Both data and programs must be loaded into main storage from input devices before they can be processed. The amount of main storage available on the system depends on the model, and, depending on the model, the amount in the configuration may be under control of model-dependent configuration controls. The storage is available in multiples of 4K-byte blocks. At any instant, the channel subsystem and all CPUs in the configuration have access to the same blocks of storage and refer to a particular block of main-storage locations by using the same absolute address.

Main storage may include a faster-access buffer storage, sometimes called a cache. Each CPU may have an associated cache. The effects, except on performance, of the physical construction and the use of distinct storage media are not observable by the program.

Expanded Storage

Expanded storage may be available on some models. Expanded storage, when available, can be accessed by all CPUs in the configuration by means of instructions that transfer 4K-byte blocks of data from expanded storage to main storage or from main storage to expanded storage. These instructions are not described. Another capability for accessing expanded storage is described in the definition of the MOVE PAGE instruction in Chapter 7, "General Instructions," and Chapter 10, "Control Instructions."

Each 4K-byte block in expanded storage is addressed by means of a 32-bit unsigned binary integer called an expanded-storage block number.

Expanded storage is not further described.

CPU

The central processing unit (CPU) is the controlling center of the system. It contains the sequencing and processing facilities for instruction execution, interruption action, timing functions, initial program loading, and other machine-related functions.

The physical implementation of the CPU may differ among models, but the logical function remains the same. The result of executing an instruction is the same for each model, providing that the program complies with the compatibility rules.

The CPU, in executing instructions, can process binary integers and floating-point numbers of fixed length, decimal integers of variable length, and logical information of either fixed or variable length. Processing may be in parallel or in series; the width of the processing elements, the multiplicity of the shifting paths, and the degree of simultaneity in performing the different types of arithmetic differ from one CPU to another without affecting the logical results.

Instructions which the CPU executes fall into five classes: general, decimal, floating-point, control, and I/O instructions. The general instructions are used in performing binary-integer-arithmetic operations and logical, branching, and other

nonarithmetic operations. The decimal instructions operate on data in the decimal format, and the floating-point instructions on data in the floating-point format. The privileged control instructions and the I/O instructions can be executed only when the CPU is in the supervisor state; the semiprivileged control instructions can be executed in the problem state, subject to the appropriate authorization mechanisms.

To perform its functions, the CPU may use a certain amount of internal storage. Although this internal storage may use the same physical storage medium as main storage, it is not considered part of main storage and is not addressable by programs.

The CPU provides registers which are available to programs but do not have addressable representations in main storage. They include the current program-status word (PSW), the general registers, the floating-point registers, the control registers, the access registers, the prefix register, and the registers for the clock comparator and the CPU timer. Each CPU in an installation provides access to a time-of-day (TOD) clock, which may be local to that CPU or shared with other CPUs in the installation. The instruction operation code determines which type of register is to be used in an operation. See Figure 2-2 on page 2-5 for the format of those registers.

PSW

The program-status word (PSW) includes the instruction address, condition code, and other information used to control instruction sequencing and to determine the state of the CPU. The active or controlling PSW is called the current PSW. It governs the program currently being executed.

The CPU has an interruption capability, which permits the CPU to switch rapidly to another program in response to exceptional conditions and external stimuli. When an interruption occurs, the CPU places the current PSW in an assigned storage location, called the old-PSW location, for the particular class of interruption. The CPU fetches a new PSW from a second assigned storage location. This new PSW determines the next program to be executed. When it has finished processing the interruption, the interrupting program may reload the old PSW, making it again the current PSW, so that the interrupted program can continue.

There are six classes of interruption: external, I/O, machine check, program, restart, and supervisor call. Each class has a distinct pair of old-PSW and new-PSW locations permanently assigned in real storage.

General Registers

Instructions may designate information in one or more of 16 general registers. The general registers may be used as base-address registers and index registers in address arithmetic and as accumulators in general arithmetic and logical operations. Each register contains 32 bits. The general registers are identified by the numbers 0-15 and are designated by a four-bit R field in an instruction. Some instructions provide for addressing multiple general registers by having several R fields. For some instructions, the use of a specific general register is implied rather than explicitly designated by an R field of the instruction.

For some operations, two adjacent general registers are coupled, providing a 64-bit format. In these operations, the program must designate an even-numbered register, which contains the leftmost (high-order) 32 bits. The next higher-numbered register contains the rightmost (low-order) 32 bits.

In addition to their use as accumulators in general arithmetic and logical operations, 15 of the 16 general registers are also used as base-address and index registers in address generation. In these cases, the registers are designated by a four-bit B field or X field in an instruction. A value of zero in the B or X field specifies that no base or index is to be applied, and, thus, general register 0 cannot be designated as containing a base address or index.

Floating-Point Registers

Four floating-point registers are available for floating-point operations. They are identified by the numbers 0, 2, 4, and 6 and are designated by a four-bit R field in floating-point instructions. Each floating-point register is 64 bits long and can contain either a short (32-bit) or a long (64-bit) floating-point operand. A short operand occupies the leftmost bit positions of a floating-

Each serial-I/O interface consists of two optical-fiber conductors between any two of a channel subsystem, a dynamic switch, and a control unit. A dynamic switch can be connected by means of multiple serial-I/O interfaces to either the same or different channel subsystems and to multiple control units. The number of control units which can be connected on one channel path depends on the channel-subsystem and dynamic-switch capabilities. Up to 256 devices can be attached to each control unit that uses the serial-I/O interface, depending on the control unit. The serial-I/O interface is described in the publication *ESA/390 ESCON I/O Interface*, SA22-7202.

I/O Devices and Control Units

I/O devices include such equipment as printers, magnetic-tape units, direct-access-storage devices, displays, keyboards, communications controllers, teleprocessing devices, and sensor-based equipment. Many I/O devices function with an external medium, such as paper or magnetic tape. Other I/O devices handle only electrical signals, such as those found in displays and communications networks. In all cases, I/O-device operation is regulated by a control unit that provides the logical and buffering capabilities necessary to operate the associated I/O device. From the programming point of view, most control-unit functions merge with I/O-device functions. The control-unit function may be housed with the I/O device or in the CPU, or a separate control unit may be used.

Operator Facilities

The operator facilities provide the functions necessary for operator control of the machine. Associated with the operator facilities may be an operator-console device, which may also be used as an I/O device for communicating with the program.

The main functions provided by the operator facilities include resetting, clearing, initial program loading, start, stop, alter, and display.

SS Format

Op Code	L	B_1	D_1	B_2	D_2
0	8	16	20	32	36 47

Op Code	L_1	L_2	B_1	D_1	B_2	D_2
0	8	12	16	20	32	36 47

Op Code	R_1	R_3	B_1	D_1	B_2	D_2
0	8	12	16	20	32	36 47

SSE Format

Op Code	B_1	D_1	B_2	D_2
0	16	20	32	36 47

Figure 5-1 (Part 2 of 2): Basic Instruction Formats

Some instructions contain fields that vary slightly from the basic format, and in some instructions the operation performed does not follow the general rules stated in this section. All of these exceptions are explicitly identified in the individual instruction descriptions.

Those instruction formats which are unique to instructions associated with the vector facility are described in the publication *IBM Enterprise Systems Architecture/390 Vector Operations*, SA22-7207.

The format names indicate, in general terms, the classes of operands which participate in the operation:

- E denotes an operation using implied operands and having an extended op-code field.
- RR denotes a register-and-register operation.
- RRE denotes a register-and-register operation having an extended op-code field.
- RX denotes a register-and-indexed-storage operation.
- RS denotes a register-and-storage operation.
- SI denotes a storage-and-immediate operation.
- S denotes an operation using an implied operand and storage.
- SS denotes a storage-and-storage operation.
- SSE denotes a storage-and-storage operation having an extended op-code field.

The first byte or, in the E, RRE, S, and SSE formats, the first two bytes of an instruction contain the op code. For some instructions in the S format, all or a portion of the second byte is ignored.

The first two bits of the first or only byte of the op code specify the length and format of the instruction, as follows:

Bit Positions 0-1	Instruction Length (in Halfwords)	Instruction Format
00	One	E/RR
01	Two	RX
10	Two	RRE/RS/RX/S/SI
11	Three	SS/SSE

In the format illustration for each individual instruction description, the op-code field shows the op code as hexadecimal digits within single quotes. The hexadecimal representation uses 0-9 for the binary codes 0000-1001 and A-F for the binary codes 1010-1111.

The remaining fields in the format illustration for each instruction are designated by code names, consisting of a letter and possibly a subscript number. The subscript number denotes the operand to which the field applies.

Register Operands

In the RR, RRE, RX, and RS formats, the contents of the register designated by the R_1 field are called the first operand. The register containing the first operand is sometimes referred to as the "first-operand location," and sometimes as "register R_1." In the RR and RRE formats, the R_2 field designates the register containing the second operand, and the R_2 field may designate the same register as R_1. In the RS format, the use of the R_3 field depends on the instruction.

The R field designates a general or access register in the general instructions, a general register in the control instructions, and a floating-point register in the floating-point instructions. However, in the instructions EXTRACT STACKED REGISTERS and LOAD ADDRESS EXTENDED, the R field designates both a general register and an access register. In the instructions LOAD CONTROL and STORE CONTROL, the R field designates a control register. (This paragraph refers only to register

sequence. Thus, it is possible for an instruction to modify the next succeeding instruction in storage.

Operations in the access-register mode or home-space mode are the same as in the other translation modes, with one exception: an instruction that is a store-type operand of a preceding instruction may appear to be fetched before the store occurs. Thus, it is not assured that an instruction can modify the succeeding instructions. This exception applies if either the storing instruction or the instruction stored is executed in the access-register or home-space mode.

Regardless of the translation mode, there are two other cases in which the copies of prefetched instructions are not necessarily discarded: (1) when the fetch and the store are done by means of different effective addresses that map to the same real address, and (2) when the store is caused by the execution of a vector-facility instruction. The case involving different effective addresses is described in more detail in "Interlocks for Virtual-Storage References" on page 5-71.

Overlapped Operation of Instruction Execution

In simple models in which operations are not overlapped, the conceptual and actual sequences are essentially the same. However, in more complex machines, overlapped operation, buffering of operands and results, and execution times which are comparable to the propagation delays between units can cause the actual sequence to differ considerably from the conceptual sequence. In these machines, special circuitry is employed to detect dependencies between operations and ensure that the results obtained, as observed by the CPU which generates them, are those that would have been obtained if the operations had been performed in the conceptual sequence. However, other CPUs and channel programs may, unless otherwise constrained, observe a sequence that differs from the conceptual sequence.

Divisible Instruction Execution

It can normally be assumed that the execution of each instruction occurs as an indivisible event. However, in actual operation, the execution of an instruction consists in a series of discrete steps. Depending on the instruction, operands may be fetched and stored in a piecemeal fashion, and some delay may occur between fetching operands and storing results. As a consequence, intermediate or partially completed results may be observable by other CPUs and by channel programs.

When a program interacts with the operation on another CPU, or with a channel program, the program may have to take into consideration that a single operation may consist in a series of storage references, that a storage reference may in turn consist in a series of accesses, and that the conceptual and observed sequences of these accesses may differ.

Storage references associated with instruction execution are of the following types: instruction fetches, ART-table and DAT-table fetches, and storage-operand references. For the purpose of describing the sequence of storage references, accesses to storage in order to perform ASN translation, PC-number translation, tracing, and the linkage-stack stacking and unstacking processes are considered to be storage-operand references.

Programming Note: The sequence of execution of a CPU may differ from the simple conceptual definition in the following ways:

- As observed by the CPU itself, instructions may appear to be prefetched in the access-register or home-space mode regardless of whether the mode exists at the time of the conceptual store or during the execution of the prefetched instruction. They may also appear to be prefetched because of a vector-facility store or when different effective addresses are used. (See "Interlocks for Virtual-Storage References" on page 5-71.)

- As observed by other CPUs and by channel programs, the execution of an instruction may appear to be performed as a sequence of piecemeal steps. This is described for each type of storage reference in the following sections.

- As observed by other CPUs and by channel programs, the storage-operand accesses associated with one instruction are not necessarily performed in the conceptual sequence. (See "Relation between Operand Accesses" on page 5-80.)

- As observed by channel programs, in certain unusual situations, the contents of storage may appear to change and then be restored to the original value. (See "Storage Change

The interruption supplies the program with information about the extent of the damage and the location and nature of the cause. Equipment malfunctions, errors, and other situations which can cause machine-check interruptions are referred to as machine checks.

Machine-Check Detection

Machine-check-detection mechanisms may take many forms, especially in control functions for arithmetic and logical processing, addressing, sequencing, and execution. For program-addressable information, detection is normally accomplished by encoding redundancy into the information in such a manner that most failures in the retention or transmission of the information result in an invalid code. The encoding normally takes the form of one or more redundant bits, called check bits, appended to a group of data bits. Such a group of data bits and the associated check bits are called a checking block. The size of the checking block depends on the model.

The inclusion of a single check bit in the checking block allows the detection of any single-bit failure within the checking block. In this arrangement, the check bit is sometimes referred to as a "parity bit." In other arrangements, a group of check bits is included to permit detection of multiple errors, to permit error correction, or both.

For checking purposes, the contents of the entire checking block, including the redundancy, are called the checking-block code (CBC). When a CBC completely meets the checking requirements (that is, no failure is detected), it is said to be valid. When both detection and correction are provided and a CBC is not valid but satisfies the checking requirements for correction (the failure is correctable), it is said to be near-valid. When a CBC does not satisfy the checking requirements (the failure is uncorrectable), it is said to be invalid.

Correction of Machine Malfunctions

Four mechanisms may be used to provide recovery from machine-detected malfunctions: error checking and correction, CPU retry, channel-subsystem recovery, and unit deletion.

Machine failures which are corrected successfully may or may not be reported as machine-check interruptions. If reported, they are system-recovery conditions, which permit the program to note the cause of CPU delay and to keep a log of such incidents.

Error Checking and Correction

When sufficient redundancy is included in circuitry or in a checking block, failures can be corrected. For example, circuitry can be triplicated, with a voting circuit to determine the correct value by selecting two matching results out of three, thus correcting a single failure. An arrangement for correction of failures of one order and for detection of failures of a higher order is called error checking and correction (ECC). Commonly, ECC allows correction of single-bit failures and detection of double-bit failures.

Depending on the model and the portion of the machine in which ECC is applied, correction may be reported as system recovery, or no report may be given.

Uncorrected errors in storage and in the storage key may be reported, along with a failing-storage address, to indicate where the error occurred. Depending on the situation, these errors may be reported along with system recovery or with the damage or backup condition resulting from the error.

CPU Retry

In some models, information about some portion of the state of the machine is saved periodically. The point in the processing at which this information is saved is called a checkpoint. The information saved is referred to as the checkpoint information. The action of saving the information is referred to as establishing a checkpoint. The action of discarding previously saved information is called invalidation of the checkpoint information. The length of the interval between establishing checkpoints is model-dependent. Checkpoints may be established at the beginning of each instruction or several times within a single instruction, or checkpoints may be established less frequently.

Subsequently, this saved information may be used to restore the machine to the state that existed at the time when the checkpoint was

IBM

Dictionary of Computing

▼ The most comprehensive computing dictionary ever published

▼ More than 18,000 entries

Limitation of Liability

While the Editor and Publisher of this book have made reasonable efforts to ensure the accuracy and timeliness of the information contained herein, neither the Editor nor the Publisher shall have any liability with respect to loss or damage caused or alleged to be caused by reliance on any information contained herein.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 9 9 8 7 6 5 4 3

ISBN 0-07-031488-8 (HC)
ISBN 0-07-031489-6 (PBK)

The sponsoring editor for this book was Daniel A. Gonneau and the production supervisor was Thomas G. Kowalczyk.

Printed and bound by R. R. Donnelley & Sons Company.

Tenth Edition (August 1993)

This is a major revision of the *IBM Dictionary of Computing,* SC20-1699-8, which is made obsolete by this edition. Changes are made periodically to the information provided herein.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Comments may be addressed to IBM Corporation, Department E37/656, P. O. Box 12195, Research Triangle Park, NC 27709.

International Edition

When ordering this title, use ISBN 0-07-113383-6.

This book is printed on acid-free paper.

and software
nal structure of
hardware and
cube, parallel

computer-aided

Synonym for

m for elec-

for computer-

cludes people,
ovide informa-
vith data proc-

communication
nother to enter
via intercon-
ence call, tele-

rity, a crime
r data. (T)
of software or

security, the
computer to
otect informa-
nformation.

system, con-
machines for

onym for

y, a computer
esentation or
something of
2) Deception
practiced in
A)

istorical clas-
on the tech-
example, first
n tubes, the
ated circuits.

computer graphics (1) Methods and techniques for converting data to or from graphic display via computers. (T) (2) That branch of science and technology concerned with methods and techniques for converting data to or from visual presentation, using computers. (A) (3) See also coordinate graphics, fixed-image graphics, interactive graphics, passive graphics, raster graphics.

computer input microfilm (CIM) A reversal of the digital-to-microimage process. Mixed media such as illustrations, graphics, and text can be digitized by a variety of methods and condensed into OCR-font CIM, then entered into the computer through an OCR workstation to update or create a document. See also optical character recognition (OCR).

computer instruction Synonym for machine instruction. (T)

computer instruction code Synonym for instruction code. (T)

computer instruction set A complete set of the operators of the instructions of a computer together with a description of the types of meanings that can be attributed to their operands. (A) Synonymous with machine instruction set.

computer-integrated manufacturing (CIM) The integration of computer operations, communications, and organizational functions for total factory automation.

Note: CIM comprises the information-technological cooperation between CAD, CAP, CAM, CAQ, and PPS. (T)

computerization Automation by means of computers. (T) (A)

computerize To automate by means of computers. (T)

computerized branch exchange (CBX) An exchange in which a central node acts as a high-speed switch to establish direct connections between pairs of attached nodes.

computer language Synonym for machine language.

computer micrographics Methods and techniques for converting data to or from microform with the assistance of a computer. (I) (A)

computer-name In COBOL, a system-name that identifies the computer on which a program is to be compiled or run.

computer network (1) A network of data processing nodes that are interconnected for the purpose of data communication. (T) (2) A complex consisting of two or more connected computing units. (A)

computer numerical control (CNC) A technique in which a machine-tool control uses a computer to store numerical-control instruction which has been generated by CAD/CAM for controlling the machine. (T)

computer operation (1) Synonym for machine operation. (T) (2) One of the elementary operations that a computer is designed to perform. (A) Synonymous with machine operation.

computer-oriented language (1) A programming language that reflects the structure of a given computer or that of a given class of computers. (I) (A) Synonymous with low-level language. (2) A programming language whose words and syntax are designed for use on a specific class of computers. (A) Synonymous with computer-dependent language, machine-oriented language. (3) See also computer language.

computer output microfilm (COM) Microfilm that contains data recorded directly from computer-generated signals. (T)

Note: The abbreviation COM is also used for the technique (computer output microfilming) and for the device (computer output microfilmer).

computer output microfilm (COM) printer A page printer that produces a micro-image of each page on a photographic film. (T)

computer output microfilmer A device for computer output microfilming. (T) (A)

computer output microfilming (COM) A technique for converting and recording data from a computer directly to a microfilmer. (A)

computer program A sequence of instructions suitable for processing by a computer. Processing may include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as the execution of the program. The sequence of instructions may include statements and necessary declarations. (T) (A)

computer program annotation Synonym for comment.

computer program origin The address assigned to the initial storage location of a computer program in main storage. (A)

computer resource Synonym for resource. (T)

instantiation A formula or pattern rule having its variables replaced by constants. (T)

instant jump A feature of some videodisc players that permits branching between frames within certain minimum distances, usually one to 200 frames away. The branch occurs during the vertical blanking interval between images.

in-stream procedure A set of job control statements placed in the input stream that can be used any number of times during a job by naming the procedure in an execute (EXEC) statement.

instruction (1) A language construct that specifies an operation and identifies its operands, if any. (T) (2) A statement that specifies an operation to be performed by a system and that identifies data involved in the operation. (3) In COBOL and Pascal, one or more clauses, the first of which starts with a keyword that identifies the instruction. Instructions affect the flow of control, provide services to the programmer, or both.

instruction address (1) The address of an instruction word. (I) (A) (2) The address that must be used to fetch an instruction. (A) (3) Contrast with address part.

instruction address register (IAR) (1) A special-purpose register used to hold the address of the next instruction to be executed. Synonymous with program register, instruction pointer register. (T) (2) A register in a processor that contains the address of the next instruction to be performed. (3) See also instruction counter, program counter.

instruction address stop An instruction address that, when fetched, causes execution to stop.

instructional design The field of education that studies the methodology of creating tools, such as computer programs, for enhancing the learning process.

instruction code (1) A code for representing the machine instructions of a computer. (T) (2) See computer instruction code. (A)

instruction control unit In a processing unit, the part that retrieves instructions in proper sequence, interprets each instruction, and applies the proper signals to the arithmetic and logic unit and other parts in accordance with this interpretation. (I) (A)

instruction counter (1) A counter that indicates the location of the next computer instruction to be interpreted. (A) (2) See also instruction address register, program counter.

instruction element (IE) A part of a processor that executes some instructions and generates operand addresses and instruction requests. It also controls the sequencing of instructions through the machine and is usually controlled by microcode.

instruction fetch The act of getting an instruction from storage and loading it into the correct registers.

instruction format The layout of the constituent parts of an instruction. (T)

instruction marker control In dictation equipment, a device used to indicate on an index slip or on the recording medium the position at which an instruction is given. (I)

instruction modifier A word or part of a word that is used to alter an instruction. (I) (A)

instruction pointer In System/38, a pointer that provides addressability for a machine interface instruction in a program.

instruction pointer register Synonym for instruction address register. (T)

instruction register (1) A register used to hold an instruction for interpretation. (I) (A) (2) See control instruction register.

instruction repertoire (1) A complete set of the operators of the statements of a computer programming language, together with a description of the types and meanings that can be attributed to their operands. (A) (2) Loosely, an instruction set. (A)

instructions In SAA Basic Common User Access architecture, text on a panel that tells a user how to interact with a panel and how to continue with the application.

instruction set (1) The set of instructions of a computer, of a programming language, or of the programming languages in a programming system. (I) (A) (2) See computer instruction set.

instruction statement See instruction *(1)*.

instruction time (I-time) The time during which an instruction is fetched from the main storage of a computer into an instruction register. See also execution time.

instruction word A word that represents an instruction. (I) (A)

INT (1) Interior. (2) Internal trace table.

intake rollers In a duplicator, paired rollers that transport paper from the paper feed mechanism to the cylinders. (T)

integer (1) One of the numbers zero, +1, -1, +2, -2... (I) (A) Synonymous with integral number. (2) A positive or negative whole number, that is, an optional sign followed by a number that does not contain a decimal place or zero. (3) In COBOL, a numeric literal or a numeric data item that does not include any digit position to the right of the assumed decimal point When the term "integer" appears in general formats, the integer must not be a numeric data item, and must not be signed, nor zero unless explicitly allowed by the rules of that format.

integer constant A string of decimal digits containing no decimal point.

integer expression An arithmetic expression with only integer type values.

integer programming (1) In operations research, a class of procedures for locating the maximum or minimum of a function subject to constraints, where some or all variables must have integer values. (I) (A) Synonymous with discrete programming. (2) Contrast with convex programming, dynamic programming, linear programming, mathematical programming, nonlinear programming, quadratic programming.

integer type An arithmetic data type that consists of integer values.

integral boundary (1) A location in main storage at which a fixed-length field, such as a halfword or doubleword, must be positioned. The address of an integral boundary is a multiple of the length of the field, expressed in bytes. See also boundary alignment. (2) In PL/I, the multiple of any 8-bit unit of information on which data can be aligned.

integral number Synonym for integer.

integral object In the C language, a character object, an object having an enumeration type, or an object having the type short, int, long, unsigned short, unsignedint, or unsigned long.

integrated Pertaining to a feature that is part of a device. Synonymous with built-in.

integrated adapter An integral part of a processing unit that provides for direct connection of a device and uses neither a control unit nor the standard I/O interface. See also integrated communication adapter, integrated file adapter.

integrated attachment An attachment that is an integral part of the basic hardware.

integrated catalog facility In the Data Facility Product (DFP), a facility that provides for integrated catalog facility catalogs.

integrated catalog facility catalog In the Data Facility Product (DFP), a catalog that consists of a basic catalog structure, which contains information about VSAM and non-VSAM data sets, and at least one VSAM volume data set, which contains information about VSAM data sets only.

integrated circuit (IC) (1) A small piece of semiconductive material that contains interconnected miniaturized electronic circuits. Synonymous with microchip, chip. (T) (2) A combination of connected circuit elements inseparably associated on or within a continuous substrate. See hybrid integrated circuit, monolithic integrated circuit.

integrated circuit memory (IC memory) A storage device composed of transistors, diodes, and other circuit elements, all fabricated on a chip of crystalline material. (T)

integrated communication adapter (ICA) A communication adapter that is an integral part of the host processor. Contrast with external communication adapter.

integrated computing The concurrent use of two or more software applications that share data; for example, word processing and computer graphics, spreadsheets and file management.

integrated database A database that has been consolidated to eliminate redundant data.

integrated digital network (IDN) A public digital end-to-end telecommunication network that is the digital backbone for the telephone network and provides multiple services, including data and leased-line transparent services. See also integrated services digital network (ISDN).

integrated disk In the programmable store system, an integral part of the store controller that is used for magnetically storing files, application programs, controller storage contents, and diagnostics.

integrated emulator An emulator program whose execution is controlled by an operating system in a multiprogramming environment. Contrast with stand-alone emulator.

integrated file adapter An integrated adapter that allows connection of multiple disk storage devices to a processing unit.

lock file (1) In a shared DASD environment under VSE, a system file on disk used by the sharing systems to control their access to shared data. (2) In AIX multiprocess applications, a system file on a disk that the sharing processes use to control their access to shared data or devices.

lock hierarchy In DPPX, a protective measure to prevent deadlocks between threads requesting locks. The locks are granted in lowest to highest order, based on their location in the hierarchy.

locking (1) A characteristic of code extension characters that applies any change in interpretation to all coded representations following, or to all coded representations of a given class, until the next appropriate code extension character occurs. (2) In DPPX, a method of ensuring uninterrupted use of a data area or code by one thread. (3) Contrast with nonlocking.

locking device On dictation equipment, a device used to fix in position a head support arm, playback head, or other critical component during transport of the equipment. (I)

lock management In IMS/VS, the reservation of a segment by a program to prevent other programs from using the segment until the program using it is done. See global lock management, local lock management.

lock mode In ACF/TCAM, a mode in which the next message received by an external logical unit (LU) entering an inquiry message for an application program is a reply from the application program to that inquiry. See also conversational mode, extended lock mode, line lock, message lock mode, station lock.

lockout (1) In a telephone circuit controlled by an echo-suppressor, the inability of one or both subscribers to get through either because of excessive local circuit noise or continuous speech from one subscriber. (2) On a calculator, the facility that inhibits entry of data when the machine is in overflow or error condition. (T) (3) In multiprocessing, a programming technique used to prevent access to critical data by both processing units at the same time. (4) To place unaddressed terminals on a multipoint line in control state so that they will not receive transmitted data. See also blind, polling, selection. (5) Synonym for protection.

lock-out facility The facility that inhibits the entry of data into a calculator when the calculator is in overflow or in error condition. (T) (A)

lock state In the AS/400 system and System/38, a condition defined for an object that determines how it is locked, how it is used (read or write), and whether the object can be shared (used by more than one job).

lock/unlock facility In OS/VS2, a supervisor facility that controls the execution of instruction strings when a disabled page fault occurs.

log (1) In ACF/TCAM, a collection of messages or message segments placed in an auxiliary storage device for accounting or data collection purposes. (2) To record; for example, to log all messages on the system printer. (3) In video production, a record of each take, shot, and scene produced; used as a guide for the editing of the tape. (4) Synonym for journal.

logarithmic axis In GDDM, an axis on which ascending powers of 10 are equally spaced.

log data set A data set consisting of the messages or message segments recorded on auxiliary storage by the ACF/TCAM logging facility.

logged-on operator A NetView operator station task that requires a terminal and a logged-on user. Contrast with autotask.

logger (1) A functional unit that records events and physical conditions, usually with respect to time. (I) (A) (2) A device that enables a user entity to log in; for example, to identify itself, its purpose, and time of entry, and to log out with the corresponding data so that the appropriate accounting procedures can be carried out in accordance with the operating system. (A)

logger task In NCCF, a subtask that records errors from EP mode and local mode devices to the EP database and transmits errors from network control program mode devices supported by VTAM and ACF/TCAM to the network control program database.

logging (1) The recording of data about specific events. (2) See data logging.

logging service facility In ACF/TCAM, a service facility that selectively causes incoming or outgoing messages or message segments to be copied onto tape or disk. The log produced by the logging service facility provides a record of message traffic through the message control program.

logic (1) The systematized interconnection of digital switching functions, circuits, or devices. (2) See double rail logic, formal logic, symbolic logic.

logical (1) Pertaining to content or meaning as opposed to location or actual implementation. (A) (2) Pertaining to a view or description of data that does not depend on the characteristics of the computer system or of the physical storage. (A) (3) Contrast with physical. (A)

logical access con

logical access cont of information-relate rather than physica control. Contrast wi

logical add Synony

logical address (1 System, an address time of manufactu address by the app the IBM 8100 Info that is either suppl during fetching and is used as a pointer

logical block In DI essarily contiguous records. A logical is transferred betw storage when an I/O

logical channel In channel and a recei to send and receive time. Several logic the same data link packets.

logical channel ide of a packet that as switched virtual circ

logical character editing symbol, non CP to delete it and acter from the input keyed in consecutiv characters are delet may be redefined o user. Synonymous

logical child In a segment that estab physical parent s segment.

Note: A logical cl its physical parent parent.

logical circuit Dep

logical comparison discover whether th

logical connection control program description of a ter

M

M Mega; 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power; 1,048,576 in decimal notation.

m (1) Milli; one thousandth part. (2) Merge order. (3) Meter.

MAC (1) Medium access control. (2) Message authentication code. (3) Mandatory access control.

MAC frame A transmission frame that controls the operation of the IBM Token-Ring Network and any ring station operations that affect the ring.

machine See accounting machine, electrical accounting machine, Turing machine, universal Turing machine.

machine address Deprecated term for absolute address *(1)*.

machine characteristic A value defined in the computer.

machine check An error condition that is caused by an equipment malfunction.

machine check analysis and recording In VSE, a feature that records machine check error information and then attempts to recover from the error.

machine check handler (MCH) A feature that analyzes errors and attempts recovery by retrying the failing instruction. If retry is unsuccessful, it attempts to correct the malfunction or to isolate the affected task.

machine check interruption (MCI) An interruption that occurs as a result of an equipment malfunction or error.

machine code Synonym for instruction code. (T)

machine configuration record In System/38, a series of data fields, modifiable only by the customer service representative, that describes the hardware.

machine cycle The shortest period of time required to execute an instruction.

machine execution priority In System/38, the priority of a routing step when competing with other routing steps for machine resources.

machine handle In a duplicator, a control for manually cycling the machine. (T) Synonymous with machine handwheel.

machine handwheel Synonym for machine handle.

machine-independent Pertaining to procedures or programs created without regard for the actual devices that are used to process them.

machine information code See system reference code.

machine instruction (1) An instruction that can be directly executed by a processor of a computer. A machine instruction is an element of a machine language. (T) (2) An instruction of a machine language. (3) Synonymous with computer instruction.

machine instruction set Synonym for computer instruction set.

machine interface (MI) (1) The means by which a device, program, user, or system interacts with a computer; for example, commands, instructions, display pointers, menus, keyboards. (2) In System/38, the instruction set and interface to the machine.

machine language (1) An artificial language composed of the machine instructions of a computer. (T) (2) A language that can be used directly by a computer without intermediate processing. (3) Synonymous with computer language.

machine learning The ability of a device to improve its performance based on its past performance. (I) (A)

machine object In the AS/400 system, a program object that has no defined storage form; the object is defined internally to the machine. The machine object is not available to the user. Contrast with data object.

machine operation An elementary function that a computer performs in response to a machine instruction. Synonymous with computer operation. (T)

machine-oriented language Synonym for computer-oriented language.

machine-readable Pertaining to data a machine can acquire or interpret (read) from a storage device, a data medium, or other source.

Micro Channel architecture The rules that define how subsystems and adapters use the Micro Channel bus in a computer. The architecture defines the services that each subsystem can or must provide.

microchip (1) Synonymous with integrated circuit (IC). (T) (2) A small piece of semiconductive material, usually silicon, that contains miniaturized electronic circuits. See also microprocessor.

microcircuit A combination of connected elements that are inseparably associated on or within a single continuous substrate to perform an electronic circuit function.

microcode (1) One or more microinstructions. (2) A code, representing the instructions of an instruction set, that is implemented in a part of storage that is not program-addressable. (3) To design, write, and also to test one or more microinstructions. (4) See also microprogram.

Note: The term microcode represents microinstructions used in a product as an alternative to hard-wired circuitry to implement functions of a processor or other system component. The term microprogram means a dynamic arrangement of one or more groups of microinstructions for execution to perform a certain function. (5) See also licensed internal code.

microcoding Coding with the use of microinstructions.

microcomputer (1) A digital computer whose processing unit consists of one or more microprocessors, and includes storage and input/output facilities. (T) (2) A small computer that includes one or more input/output units and sufficient memory to execute instructions; for example, a personal computer. The essential components of a microcomputer are often contained within a single enclosure. See also laptop computer, mainframe, minicomputer, personal computer, supermini.

Notes:

1. *Microcomputers are usually desk-top or portable devices with a display, a keyboard, and tape, disk, and diskette storage; they are designed primarily for stand-alone operation but can be used as workstations in terminal emulation mode.*
2. *In terms of size and processing power, the hierarchy of computers consists of supercomputers, mainframes (usually called processing units or processors), superminis, minicomputers, and microcomputers. As the computing power and storage capability of microcomputers grow and the size of minicomputers decreases to table-top dimensions, the distinctions between micros and minis will become less distinct and may eventually disappear.*

microdiagnostic utility In the Data Facility Hierarchical Storage Manager, a program run by a service representative to test a machine.

microfiche (1) A sheet of microfilm capable of containing microimages in a grid pattern, usually containing a title that can be read without magnification. (A) (2) See also ultrafiche.

microfilm (1) A high resolution film for recording microimages. (A) (2) To record microimages on film. (A) (3) Microform whose medium is film, in the form of rolls, that contains microimages arranged sequentially. (4) See computer output microfilm.

microfilmer See computer output microfilmer.

microfilming See computer output microfilming. (A)

microform A medium that contains microimages; for example, microfiche, microfilm. (T) (A)

microform reader A device that enlarges microimages for viewing.

microform reader-copier A device that performs the functions of a reader and a printer to produce hard copy enlargements of selected microimages. Synonymous with microform reader-printer.

microform reader-printer Synonym for microform reader-copier.

microfreeze A machine state in which a central processor is stopped. Microfreeze is initiated by setting the microfreeze bit on in a selected control word. Thereafter, when that control word is sensed, the central processor stops in the microfreeze state.

micrographics (1) The branch of science and technology concerned with techniques for converting any form of information to or from microform. (A) (2) See computer micrographics.

microimage An image too small to be read without magnification. (A)

microinstruction An instruction for operations at a level lower than machine instructions. (T)

micrometer One millionth part of a meter. Synonymous with micron.

micron Synonym for micrometer.

micro order A part of a microcoded control word that controls specific machine operations.

microprocessor (1) A processor whose elements have been miniaturized into one or a few integrated circuits. (T) (2) A microchip containing integrated circuits that executes instructions. See also microcomputer.

microprogram (1) A sequence of microinstructions. Microprograms are mainly used to implement machine instructions. (T) (2) A group of microinstructions that when executed performs a preplanned function.

Note: The term microprogram represents a dynamic arrangement or selection of one or more groups of microinstructions for execution in order to perform a particular function. The term microcode represents microinstructions used in a product as an alternative to hard-wired circuitry to implement certain functions of a processor or other system component.

microprogram load See initial microprogram load.

microprogramming (1) The preparation of microprograms. (2) The technique used in the design of hardware that is to be controlled by microprograms. (T)

microsecond One-millionth of a second.

microwave Any electromagnetic wave in the radio frequency spectrum above 890 megahertz.

MID Machine identifier.

mid-batch recovery In ACF/TCAM, the ability to recover from permanent text errors encountered in any block of data following the first block in a multiple-block message.

middle-in-chain (MIC) A request unit (RU) whose request header (RH) begin chain indicator and RH end chain indicator are both off. See also RU chain.

middle letter row On a keyboard, the center letter row. Synonymous with row C. See lower letter row, upper letter row. See also numeric row.

MIDI Musical Instrument Digital Interface.

mid-shot Synonym for medium shot.

migrate (1) To move data from one hierarchy of storage to another. (2) To move to a changed operating environment, usually to a new release or version of a system.

migration (1) The process of moving data from one computer system to another without converting the data. (2) Installation of a new version or release of a program to replace an earlier version or release. (3) See data migration, general migration, interval migration, page migration.

migration cleanup In the Data Facility Hierarchical Storage Manager, the first phase of daily space management. This process deletes unnecessary records and migration copies.

migration control data set In the Data Facility Hierarchical Storage Manager, a VSAM key-sequenced data set that contains statistics records, control records, user records, records for data sets that have migrated, and records for volumes under migration control of the Data Facility Hierarchical Storage Manager.

migration data host A VTAM node that acts as both an APPN end node and a type 5 subarea node. Contrast with interchange node.

migration volume A volume under control of the Data Facility Hierarchical Storage Manager that contains migrated data sets.

MIH Missing-interrupt handler.

mike mixer In audio and video production, a device for combining input from two or more microphones or other audio sources for recording.

milli (m) One thousandth.

millibyte (mb) Deprecated term for kilobyte (Kb).

milliliter (ml) One thousandth of a liter; 0.27 fluid drams.

millimeter (mm) One thousandth of a meter; 0.04 inch.

Millions of instructions per second (MIPS) A measure of processing performance. (T)

millisecond One thousandth of a second.

mimic Synonym for impersonate.

minicomputer (1) A digital computer that is functionally intermediate between a microcomputer and a mainframe. (T) (2) An intermediate-size computer that can perform the same kinds of applications as a mainframe but has less storage capacity, processing power, and speed than a mainframe. See also mainframe, microcomputer, personal computer, supercomputer, supermini.

Note: Minicomputers are floor-standing or desktop devices, frequently connected to a mainframe to perform subsidiary operations. They are priced for purchase by smaller businesses and are sufficiently compact to be located in a typical office environment.

Note: The process interface system may be part of a special-purpose computer.

process entry In ACF/TCAM, a terminal-table entry representing an application program. A process entry must be defined for each queue to which an application program can issue a GET or READ macroinstruction. At least one process entry must be defined for all PUT and WRITE macroinstructions from the same program. See also cascade entry, group entry, line entry, logtype entry, single entry, terminal-table entry.

process exception Synonym for program exception.

process group In the AIX operating system, a grouping of processes that permits the signaling of related groups of processes. A newly created process joins the process group of its creator.

processing (1) The performance of logical operations and calculations on data, including temporary retention of data in processor storage while the data is being operated on. (2) In a document copying machine, the treatment of sensitized material after exposure so as to reveal and retain the image. (T) (3) The action of performing operations on input data.

processing and control element (PCE) In the 8100 Information System, the part of the processor that contains sequencing and processing controls for instruction execution, interruption control, dynamic address translation, and other control and processing functions.

processing intent In IMS/VS, an application program attribute defined in the program communication block (PCB) that specifies the program's database access privileges, such as insert, delete, and replace.

processing level In System/36, a stage in the work-station utility program cycle.

processing limit In IMS/VS, a transaction attribute that defines how many messages the application program can process during one program execution.

processing program (1) A program that performs such functions as compiling, assembling, or translating for a particular programming language. (2) Any program capable of operating in the problem program state. This includes IBM-distributed language processors, application programs, service programs, and user-written programs.

processing services In DPCX, the part of program services that provides data processing capabilities.

processing system See data processing system. (A)

processing unit (1) A functional unit that consists of one or more processors and their internal storages. (I) (A) (2) See primary processing unit, processor. (3) See also mainframe, processor complex. See Figure 117.

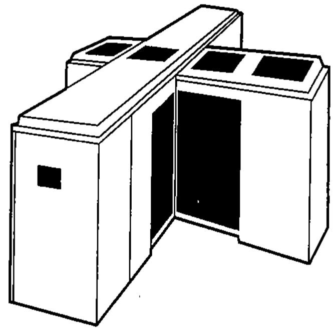


Figure 117. Processing Unit

process interface system A functional unit that adapts process control equipment to the computer system in a process computer system. (T)

process interrupt signal A signal that originates from a technical process and that causes an interrupt in the process computer system. (T)

process lock In the AIX operating system, a lock that allows the calling process to lock or unlock both its text and data segments into memory.

processor (1) In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (T) (2) One or more integrated circuits that process coded instructions and perform a task. See also system processor, service processor, input/output processor. (3) Deprecated term for processing program.

processor address space In the IBM 8100 Information System, the set of relocated addresses numbered sequentially from zero to the maximum available address.

processor complex A configuration that consists of all the machines required for operation; for example, an IBM 3081 Processor Complex, consisting of a 3081 Processor Unit, 3082 Processor Controller, 3087 Coolant Distribution Unit, 3089 Power Unit, and a 3278 Display Console Model 2A.

processor configuration In the System/370 computing system, the ability to enable or disable storage elements for one or more processing units. In the multiprocessing mode, any storage element that is enabled for one processing unit can be accessed by

og of information about
hanges and patches to

\n integrated collection
upport the development

artificial language for
(T)

rete, identifiable set of
ıs a unit, by an assem-
loading routine, or other
(A)

ce quotation PRPQ. A
ıotation on alterations or
capabilities of system
ensed programs. The
ınction with computing
data processing prob-

p. user profile In
upplied by the Control
ıas the authority neces-
ervice representative to
nming and the special
ts and job control rights.

of a set of symbolic
ograms.

a programming environ-
or the development and
T) (2) In a data proc-
eded to use one or more

) system and System/38,
user can enter BASIC
to the system from the
of the statements are
Contrast with data mode.

L Identification Division
a user-defined word that
ogram.

OBOL, an entry in the
the Identification Divi-
at specify the program-
ogram attributes to the

AS/400 system and
ıe object classifications.
d in programs that get
object definition table.

Program objects are used as the parameter or values of machine instructions. Contrast with system object.

program offering An unwarranted licensed program. See also vendor-logo product.

program operator An VTAM application program that is authorized to issue VTAM operator commands and receive VTAM operator awareness messages. See also solicited message, unsolicited message.

program operator interface (POI) A VTAM function that allows programs to perform VTAM operator functions.

program origin See computer program origin.

program parameter See external program parameter.

program patch (1) A temporary fix that is made to a program. (2) In System/38, a method of repairing a program at the machine interface program template level.

program product Deprecated term for licensed program.

program production time That part of system production time during which a user's computer program is successfully executed. (I) (A)

program recursion level In PL/I, a count that is increased when a program or external procedure is called repeatedly. The program recursion level can be specified on the system debug commands through the RCRLVL parameter. Contrast with procedure recursion level.

program register Synonym for instruction address register. (T) (A)

program required credentials (PRC) In DPPX, the value associated with a load module that determines whether a user must be authorized to request execution of the load module.

program run The performance of one or more programs. (T) (A)

program security See data processing system security.

program selection A function that identifies and actuates a program for operation.

program selector In word processing, a control that allows selection of a machine program for operation. (T)

program-sensitive fault (1) A fault that occurs as a result of the execution of some particular sequence of instructions. (I) (A) (2) Contrast with pattern-sensitive fault.

program services In DPCX, the logical layer that translates data manipulation requests from user programs and system services to resource manipulation requests to a resource manager.

program specification A document that describes the structure and functions of a program in sufficient detail to permit programming and to facilitate maintenance. (T)

program stack (1) In the AS/400 system, a list of programs linked together as a result of programs calling other programs with the CALL instruction, or implicitly from some other event, within the same job. (2) See invocation stack.

program static storage area (PSSA) In the AS/400 system, a system object that contains static variable data for programs on the program stack. The PSSA contains space for program variables that is activated when the program object is activated. The PSSA is contained in the process access group (PAG).

program status register A register that contains conditions that can be tested by branch or jump instructions.

program status vector (PSV) In the IBM 8100 Information System, the formatted information used to control the order in which instructions are executed by the associated program. See primary PSV, secondary PSV.

program status word (PSW) An area in storage used to indicate the order in which instructions are executed, and to hold and indicate the status of the computer system. Synonymous with processor status word.

program structure The manner in which the component parts of a computer program, such as identifiers, declarations, and statements are arranged. (T)

Program Support Representative (PSR) See IBM Program Support Representative.

program table In the AS/400 system, a list of the finance applications for use in an finance job. Each table entry consists of a program ID and the program name and library associated with that ID. Program IDs received in data streams from finance devices are located in the program table to determine the AS/400 application to be called.

computer-assisted instruction system in which response to input is fast enough to affect subsequent input.

real-time control The control of a process by real-time processing. (I) (A)

real-time input Input data received into a data processing system within time limits that are determined by the requirements of some other system or at instants that are so determined. (I) (A)

real-time operation (1) In analog computing, operation in the computer mode, during which the time scale factor is one. (I) (A) (2) Synonym for real-time processing.

real-time output Output data delivered from a data processing system within time limits that are determined by the requirements of some other system or at instants that are so determined. (I) (A)

real-time processing The manipulation of data that are required or generated by some process while the process is in operation; usually the results are used to influence the process, and perhaps related processes, while it is occurring. (I) (A) Synonymous with real-time operation.

real-time simulation The operation of a simulator so that the time scale factor is equal to one for a physical time specified by the system being simulated and by the corresponding computer time of the simulator. (A)

Real-Time Video (RTV) In Digital Video Interactive (DVI) technology, a video compression technique that operates in real time using the DVI system itself. It provides picture quality suitable for application development, but the final application is usually compressed using Production Level Video (PLV).

rear compression In VSAM, the elimination from a key of characters to the right of the first character that is unequal to the corresponding character in the following key.

reason code A code that identifies the reason for a detected error.

reasonableness check A check to determine whether a value conforms to specified criteria. (T)

reassign To mark a disk sector as damaged. The marked disk sector points to another sector location where the data from the damaged sector is moved.

reboot Synonym for system reset.

rebound distance On a typewriter, the distance between the face of the type element and the platen when the type element has engaged its stop. This distance influences the force of impression and depends on the thickness of the paper. (T)

rebuild maintenance A method of maintaining keyed access paths for database files. This method updates the access path only while the file is open. After the file is closed and reopened, the access path is rebuilt. Contrast with delay maintenance, immediate maintenance.

recall (1) In ACF/TCAM, a method of retrieving a message or a part of a message in order to process or redirect it. (2) In the Data Facility Hierarchical Storage Manager, the process of moving a migrated data set from a level 1 or level 2 volume to a primary volume.

receive (1) To obtain and store data. (2) In systems with ACF/TCAM, to obtain a message transmitted from a terminal to the computer over a line. Contrast with send. See also accept, enter.

received line signal detector (RLSD) A signal defined in the EIA-232 standard that indicates to the data terminal equipment (DTE) that it is receiving a signal from the remote data circuit-terminating equipment (DCE).

receive interruption The interruption of a transmission to a terminal by a higher priority transmission from the terminal. Synonymous with break.

receive leg The side of a duplex line that is receiving. Contrast with transmit leg.

receive mode In the AS/400 system and System/38, a time during which the BSC adapter looks for synchronization characters, and stores the data characters in main storage.

receive not ready (RNR) In communications, a data link command or response that indicates a temporary condition of being unable to accept incoming frames.

receive not ready (RNR) packet See RNR packet.

receive-only typing reperforator A teletypewriter receiver producing perforated tape with characters along the tape edge. Synonymous with rotor.

receive pacing In SNA, the pacing of message units being received by a component. See also send pacing.

receiver (1) A person or thing that receives something. See also addressee. Contrast with sender. (2) See journal receiver.

receiver directory In the AS/400 system, summary information about the journal receivers that are or

IBM Terminology

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #

Please send any feedback about the terms and definitions on this site to terms@ca.ibm.com

Q

Q.921
: The ITU-T (formerly CCITT) recommendation that defines the link layer of the DSS1 protocol. Q.921 defines an HDLC protocol that ensures a reliable connection between the network and the user. Often used synonymously with LAPD.

Q.931
: An ITU recommendation that defines the network layer protocol for integrated services digital network (ISDN). This layer carries the ISDN messages that control the establishment and clearing of calls.

Q.932
: The CCITT Recommendation that defines the generic procedures applicable for the control of supplementary services at the user-network interface. These procedures expand on the basic call-control functions defined in Q.931. See also supplementary service.

QA
: See quality assurance.

Q Apply latency
: In Q replication, an approximate measurement of the difference between the time that the Q Apply program gets changed data from the receive queue and the time that the data is applied to a target table. This is a subset of the end-to-end latency in a replication scenario. See also latency, Apply latency, Capture latency, Q Capture latency, queue latency, end-to-end latency.

Q Apply program
: In Q replication, a program that reads transactions from a receive queue and applies those changes to one or more target tables or passes the changes to a procedure.

Q Apply schema
: In Q replication, the identifier for a Q Apply program and its control tables.

Q Apply server
: In Q replication, a database or subsystem on which the control tables for the Q Apply program are located and where the Q Apply program runs. It contains one or more sets of the control tables that store information about target tables and other replication definitions.

QBE
: See Query by Example.

QBIC
: See Query by Image Content.

Qbuffer
: See queue buffer.

Q Capture latency
: In Q replication, an approximate measure of how current a Q Capture program is in reading the DB2 database recovery log. It is the approximate difference between the time that source data was changed and the time that the Capture program made the data

refresh age
: The time duration between the current time and the time during which a materialized query table was last refreshed.

refresh pack
: A cumulative collection of fixes that contains new functions. See also fix pack, test fix, interim fix, manufacturing refresh, fix.

refund
: In WebSphere Commerce Payments, the credit amount in the smallest denomination of the particular currency used to place the order.

region
: (1) In MVS, a variable-size subdivision of virtual storage that is allocated to a job step or system task. CICS Transaction Server runs in an MVS/ESA region, usually referred to as the CICS region.
(2) A physical instance of a CICS server.
(3) A contiguous area of virtual storage that has common characteristics and that can be shared between processes.

region class
: The class IMS assigns to a message region that indicates the message classes that can be processed within the region. See also class, message class.

region-remote
: A term used in early releases of CICS to refer to a CICS system in another region of the same processor. It can be taken to refer to a system that is accessed through an IRC (MRO) link, rather than through an SNA LU6.1 or LU6.2 link.

region size
: The amount of main storage available for a program to run.

register
: (1) An internal computer component capable of storing a specified amount of data and accepting or transferring this data rapidly.
(2) In the hierarchical file system, to make an underlying file system and the specific functions it supports known to the application programming interface layer and accessible to user applications.
(3) To insert authorization and authentication information into binding information.
(4) In SQL replication, to define a DB2 table, view, or nickname as a replication source.
(5) To add a user-written condition handler onto a routine's stack frame.

registered customer
: (1) A customer who is registered with a store. To register, a customer provides personal information to the WebSphere Commerce system, such as an e-mail address.
(2) In WebSphere Commerce, a defined role that allows the reseller to shop in the marketplace. Resellers must first register in the marketplace and be approved by the seller administrator in order to attain the registered customer role.

registered enterprise-unique identifier
: A name given to an entire network that makes the network unique among other networks, including IBM networks. New users are requested to register the network name with IBM if they plan to communicate with IBM networks (for PTF information, for example).

registered filter
: A filter that allows more than one active filter for alerts and problem logs. When a filter is registered, the system can send notification of events to a data queue. Registered filters behave slightly different than filters exposed through the network attributes or system value commands.

registered name

ENCYCLOPEDIA OF COMPUTER SCIENCE

THIRD EDITION

Edited by
Anthony Ralston
Edwin D. Reilly

IEEE PRESS

Library of Congress Catalog Card Number 92-14553
ISBN 0–442–27679–6

Printed in the United States of America

Van Nostrand Reinhold
115 Fifth Avenue
New York, New York 10003

Chapman and Hall
2-6 Boundary Row
London SE1 8HN, England

Thomas Nelson Australia
102 Dodds Street
South Melbourne 3205
Victoria, Australia

Nelson Canada
1120 Birchmount Road
Scarborough, Ontario M1K 5G4, Canada

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Encyclopedia of computer science / Anthony Ralston, Edwin D. Reilly, editors ; Caryl Ann Dahlin, managing editor.
p. cm.
Includes bibliographical references and index.
ISBN 0-442-27679-6
1. Computer science—Encyclopedias. 2. Information science—Encyclopedias. I. Ralston, Anthony. II. Reilly, Edwin D.
QA76.15.E48 1992
004'.03—dc20 92-14553
CIP

ishes using the reentrant program, and another user process (say, UP2) gets to use the reentrant program, the reentrant program must have communicated to it the location of the data for UP2 and the place I2 where previous execution was interrupted. Reentrant programs are sometimes called *pure procedures* or *sharable code.*

J. A. N. LEE

REGISTER

For articles on related subjects *see* ARITHMETIC-LOGIC UNIT; BASE REGISTER; CENTRAL PROCESSING UNIT; GENERAL REGISTER; INDEX REGISTER; PROGRAM COUNTER; and SHIFTING.

A register is a specialized storage element of the CPU that consists of digital storage elements that respond faster than those typically used to implement main memory storage locations.

The purpose of a register is to store a string of bits (often a word) representing related information: the digits of a number, the symbols of an alphanumeric word, the bits representing the status of various parts of a computer, the bits indicating the presence of interrupt requests, etc. The bits X_i that are stored in the n-bit register X are considered to be arranged in linear order and are identified by the indices i, usually chosen in the range $0 \leq i \leq n - 1$ (Fig. 1).

All registers within a computer or other digital device are uniquely identified by names or addresses. The names (e.g. X, ACC (Accumulator), PC (Program Counter), MSW (machine status word), index register, etc. often indicate the function of a register. The addresses are a set of N consecutive integers A ($0 \leq A \leq N - 1$) which identify registers within a storage array (often called *local* memory). The number of bits that can be stored in a register is its *length.* Registers of several different lengths may be found within the same system, but the most common length is the word length of the computer.

Registers are provided with the means to *load* new words or individual bits (writing) and to *sense* the register's contents (reading). If the reading and writing operations use all bits of a register simultaneously, the register is called *parallel*, but if one bit at a time is used, the register is called *serial.* The difference affects the reaction time of the register, but not the way a programmer uses it.

Registers may be provided with other functions in addition to reading and writing. A *shift register* is a register in which all bits may be displaced by one or more positions to the left or to the right. A *counter* is a register in which the contents go through a specified sequence of states, normally those that represent consecutive binary integers. An *accumulator* is a register to which an adder circuit adds a specified number to the prior contents of the accumulator. Shifting, counting, and accumulating are performed upon receipt of appropriate machine-language commands.

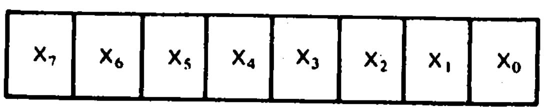


FIG. 1. An eight-bit register X.

ALGIRDAS AVIŽIENIS

REGRESSION ANALYSIS

For articles on related subjects *see* LEAST SQUARES APPROXIMATION; MATHEMATICAL PROGRAMMING; MATHEMATICAL SOFTWARE; MATRIX COMPUTATIONS; NUMERICAL ANALYSIS; SIMPLEX METHOD; and STATISTICAL APPLICATIONS.

Regression methods are used to relate a *response* variable to one or more *predictor* variables. There are three main uses of regression analysis:

1. Prediction of future values of the response variable(s), given hypothetical values of the predictor variables.
2. Specification of a model by selection of a subset of the response variables.
3. Estimation of parameters in a model given by a set of response variables.

The simplest possible model has a single predictor variable and is written

$$y_i = \beta_0 + \beta_1 x_i + e_i, \quad i = 1, \ldots, n$$

where y_i is the value of the response variable for the ith observation, x_i is the value of the predictor variable for the same observation (assumed to be fixed), β_0 and β_1 are unknown constants, and the error ϵ_i is a random variable. Typically, one assumes that the ϵ's are independent and have a Gaussian distribution with mean zero and constant variance. In this case the least-square estimates of the unknown parameters (the β's) and the maximum likelihood estimates of the unknown parameters are identical. Consequently, they have many nice statistical properties, in addition to being simple to compute. A straightforward generalization to p predictor variables is written

$$Y = X\beta + \epsilon$$

where now Y is an $n \times 1$ column vector, X is an $n \times p$ matrix of the p (fixed) predictor vectors, β is a $p \times 1$ vector of unknown parameters, and ϵ is an $n \times 1$ vector of random variables. The least squares estimates of the parameters (denoted $\hat{\beta}$) are given by the solution of the so-called *normal equations*

$$X^T X \hat{\beta} = X^T Y.$$

Standard textbooks on the subject include Draper and Smith (1981), Weisberg (1985), and Myers (1990).

register

1. One of a small number of high-speed memory locations in a computer's CPU. Registers differ from ordinary random-access memory in several respects:

There are only a small number of registers (the "register set"), typically 32 in a modern processor though some, e.g. SPARC, have as many as 144. A register may be directly addressed with a few bits. In contrast, there are usually millions of words of main memory (RAM), requiring at least twenty bits to specify a memory location. Main memory locations are often specified indirectly, using an indirect addressing mode where the actual memory address is held in a register.

Registers are fast; typically, two registers can be read and a third written -- all in a single cycle. Memory is slower; a single access can require several cycles.

The limited size and high speed of the register set makes it one of the critical resources in most computer architectures. Register allocation, typically one phase of the back-end, controls the use of registers by a compiled program.

See also accumulator, FUBAR, orthogonal, register dancing, register allocation, register spilling.

2. An addressable location in a memory-mapped peripheral device. E.g. the transmit data register in a UART.

Try this search on Wikipedia, OneLook, Google

Nearby terms: regexp « Regina « regional network « **register** » register allocation » register assignment » register dancing

---

techweb NETWORK

TechEncyclopedia More than 20,000 IT terms

Results found for: PSW

PSW

(**P**rogram **S**tatus **W**ord) A hardware register that maintains the status of the program being executed.

TERMS SIMILAR TO YOUR ENTRY

Entries before PSW

- PSNEXT
- PSP
- PSpice
- PSS
- PSTN

Entries after PSW

- psycho-acoustic model
- PTC
- PTG&LI
- PTOCA
- PTP

DEFINE ANOTHER IT TERM

Define

Or get a random definition

M6.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D C 20231
www uspto gov

APPLICATION NO	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO
09/337,158	06/21/1999	DAVID ARLEN ELKO	PO9-99-007	8602

7590 01/23/2002

BLANCHE E SCHILLER ESQ
HESLIN & ROTHENBERG P C
5 COLUMBIA CIRCLE
ALBANY, NY 12203

EXAMINER	
PARK, ILWOO	
ART UNIT	**PAPER NUMBER**
2185	4

DATE MAILED: 01/23/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

H.6

H-6

Office Action Summary	Application No.	Applicant(s)
	09/337,158	ELKO ET AL.
	Examiner	Art Unit
	Ilwoo Park	2185

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 June 1999*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 June 1999* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 2.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

DETAILED ACTION

1. Claims 1-49 are presented for examination.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless --
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

3. Claims 1, 4-13, 16, 17, 23, 26-34, 37, 39, 42, 43, 44, 45, 46, and 47 are rejected under 35 U.S.C. 102(b) as being anticipated by Frey et al., US patent No. 5,416,921.

As to claims 1, 16, 17, 37, and 39, Frey et al teach [col. 20, line 60-col. 21, line :7] a method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value [time value from system clock 60] timing information comprising at least one of time-of-day information and data information; and

including as another part [system configuration identifier] of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images [col. 2, lines 23-37] on one or more processors of said computing environment.

4. As to claims 4, 8, and 26, Frey et al teach said providing and said including are performed by an instruction [col. 20, lines 60-68].

5. As to claims 5, 10, 13, 27, 31, 34, 42, 43, and 45, Frey et al teach retrieving by an instruction said timing information from a physical clock [system clock 60] and retrieving said selected information from a storage [col. 20, line 60-col. 21, line 17].

6. As to claims 6 and 28, Frey et al teach said storage area is a programmable register [col. 20, line 60-col. 21, line 17] by a set register instruction [inherent: col. 2, lines 23-26].

7. As to claims 7, 29, and 44, Frey et al teach initializing said physical clock independently of setting said programmable register [col. 12, lines 55-68].

8. As to claims 9 and 30, Frey et al teach initializing said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value [col. 20, line 60-col. 21, line 17] .

9. As to claims 11, 32, and 46, Frey et al teach initializing said physical clock to a predetermined value using a set instruction [inherent: col. 2, lines 23-26; col. 12, lines 55-68; col. 20, line 60-col. 21, line 17].

10. As to claims 12, 33, and 47, Frey et al teach obtaining said selected information from a programmable register independent from said physical clock and said set instruction [col. 20, line 60-col. 21, line 17].

11. Claims 1, 3, 4, 16, 19, 23, 25, 26, 37, 39, and 41 are rejected under 35 U.S.C. 102(b) as being anticipated by Cwiakala et al., US patent No. 5,257,379.

As to claims 1, 16, 23, 37, and 39, Cwiakala et al teach [col. 17, line 25-col. 18, line 17] a method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part [time-of-day clock 1101E] of a sequence value timing information comprising at least one of time-of-day information and data information; and

including as another part [operating system configuration identifier 1101F] of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

12. As to claims 3, 19, 25, and 41, Cwiakala et al teach providing as a further part of said sequence value a processor identifier [col. 17, lines 48-50].

13. As to claims 4 and 26, Cwiakala et al teach said providing and said including are performed by an instruction [col. 31, lines 34-42].

Claim Rejections - 35 USC § 103

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. Claims 2, 14, 15, 18, 20, 21, 24, 35, 36, 38, 40, 48, and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Frey et al., US patent No. 5,416,921 and Sugiyama, US patent No. 5,933,625.

As claims 2, 14, 18, 20, 24, 35, 38, 40, and 48, Sugiyama teach a method of generating unique sequence values usable within a computing environment [col. 3, lines 26-50], said method comprising providing as a further part of sequence value a placeholder value [col. 9, line 63-col. 10, line 3] usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide timing information of said sequence value wraps back to zero.

Therefore, it would have been obvious to one of ordinary skill in the art to combine the teachings of Frey et al and Sugiyama at the time of the invention because they both teach a method of generating unique sequence values using a timing information from a physical clock and the Sugiyama's teaching of including as a part of the sequence value the placeholder value would further increase uniqueness of the Frey et al's sequence values [Frey et al: col. 5, line 62-col. 6, line 18].

16. As to claims 15, 21, 36, and 49, Frey et al. teach the claimed invention [see above].

17 Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Frey et al., US patent No. 5,416,921 and Sugiyama, US patent No. 5,933,625 as applied to claim 21 above, and further in view of Cwiakala et al., US patent No. 5,257,379.

As to claim 22, Cwiakala et al teach providing as a further part of said sequence value a processor identifier [col. 17, lines 48-50].

Therefore, it would have been obvious to one of ordinary skill in the art to combine the teachings of Frey et al and Sugiyama at the time of the invention to include a processor identifier in the sequence value in order to further increase uniqueness.

Conclusion

18. Any inquiry concerning this communication should be directed to Ilwoo Park, whose telephone number is (703) 308-7811 or via e-mail, ***ilwoo.park@uspto.gov***. The Examiner can normally be reached Monday through Friday from 9:00 AM to 5:30 PM.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Thomas C. Lee, can be reached at (703) 305-9717.

Any inquiry of a general nature of relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 746-7239 (for formal communications intended for entry),

(703) 746-7238 (for after-final communications),

or:

(703) 746-7240 (for informal or draft communications, please label "PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA., Sixth Floor (Receptionist)

Ilwoo Park

January 16, 2002

THOMAS LEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

PAGE 1 OF 1

FORM PTO-892	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	SERIAL NO. 09/337,158	GROUP ART UNIT 2185	ATTACHMENT TO PAPER NO.	4
NOTICE OF REFERENCES CITED		APPLICANT(S) ELKO ET AL.			

U.S. PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	NAME	CLASS	SUB-CLASS	FILING DATE
	A	5,416,921	5/1995	Frey et al	714	11	
	B	5,257,379	10/1993	Cwiakala et al	713	1	
	C	5,933,625	8/1999	Sugiyama	713	503	
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

FOREIGN PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUB-CLASS
	L						
	M						
	N						
	O						
	P						
	Q						

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, Etc.)

	R	
	S	
	T	
	U	

EXAMINER	DATE	
Ilwoo Park	January 16, 2002	Form892ccs2106b

* A copy of this reference is not being furnished with this office action.
(See Manual of Patent Examining Procedure, section 707.05(a).)

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)
(Large Entity)

Docket No.
PO9-99-007

In Re Application Of: Elko et al.

COPY OF PAPERS ORIGINALLY FILED

OIPE JUL 0 1 2002 PATENT & TRADEMARK OFFICE

Serial No.	Filing Date	Examiner	Group Art Unit
09/337,158	06/21/99	Ilwoo Park	2185

Invention: **METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES**

RECEIVED
JUL 0 9 2002
Technology Center 2100

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response to the Office Action of January 23, 2002 (*Date*) in the above-identified application.

The requested extension is as follows (check time period desired):

☐ One month ☒ Two months ☐ Three months ☐ Four months ☐ Five months

from: April 23, 2002 (*Date*) until: June 23, 2002 (*Date*) (Sunday)

The fee for the extension of time is $400 and is to be paid as follows:

☒ A check in the amount of the fee is enclosed.

☒ The Commissioner is hereby authorized to ~~charge any fees which may be required, or~~ credit any overpayment, to Deposit Account No. 08-1935
A duplicate copy of this sheet is enclosed.

☒ If an additional extension of time is required, please consider this a petition therefor and charge any additional fees which may be required to Deposit Account No. 08-1935
A duplicate copy of this sheet is enclosed.

Blanche E. Schiller
Signature

Dated: June 24, 2002

Blanche E. Schiller, Esq.
Reg. No. 35,670
HESLIN ROTHEBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

I certify that this document and fee is being deposited on June 24, 2002 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Blanche E. Schiller
Signature of Person Mailing Correspondence

Blanche E. Schiller
Typed or Printed Name of Person Mailing Correspondence

07/05/2002 RMEBRAHT 00000150 09337158
01 FC:116 400.00 OP

CC:

P12LARGE/REV06

OIPE JC48
JUL 0 1 2002
PATENT & TRADEMARK OFFICE

COPY OF PAPERS ORIGINALLY FILED

RECEIVED
JUL 0 9 2002
Technology Center 2100

#6/A
7/19/02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Elko et al.	Confirmation No.: 8602
Serial No.:	09/337,158	Group Art Unit: 2185
Filed:	06/21/99	Examiner: Ilwoo Park

Title: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

Certificate of Mailing

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on June 24, 2002.

Blanche E. Schiller
Blanche E. Schiller
Attorney for Applicants
Reg. No. 35,670

Date of Signature: June 24, 2002

Assistant Commissioner for Patents
Washington, D.C. 20231

Response to Office Action

Dear Sir:

This Response to Office Action is being submitted in reply to the Office Action mailed January 23, 2002 for the above-referenced patent application. A response to the Office Action was initially due on or before April 23, 2002, and thus, a Request For Extension Of Time and the requisite fee are enclosed herewith. Since June 23, 2002 is a Sunday, this Response is being timely filed on Monday, June 24, 2002.

Amendments

Please amend the above-identified application, as follows:

In the Specification:

Please amend page 1, paragraph 2 of the Cross-Reference to Related Applications, as follows:

A1

"METHOD, SYSTEM AND PROGRAM PRODUCTS FOR EMPLOYING EXPANDED PHYSICAL CLOCKS," by Elko et al., Serial No. 09/337,157, (Docket No. PO9-99-064), now US patent No. 6,490,689, filed June 21, 1999. 4/2/04

In the Claims:

Please cancel claim 17, without prejudice, and amend claims 1, 14-16, 18, 20-21, 23, 35-39, 48 and 49, as follows. All of the claims are reproduced for the Examiner's convenience.

Sub B1

A2

1. (AMENDED) A method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

including as another part of said sequence value selected information usable in making said sequence

A2B1

value unique across a plurality of operating system images on one or more processors of said computing environment.

2. The method of claim 1, further comprising providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

3. The method of claim 1, further comprising providing as a further part of said sequence value a processor identifier.

4. The method of claim 1, wherein said providing and said including are performed by an instruction.

5. The method of claim 4, wherein said providing comprises retrieving, by said instruction, said timing information from a physical clock, and wherein said including comprises retrieving, by said instruction, said selected information from a storage area.

6. The method of claim 5, wherein said storage area is a programmable register set by a set register instruction.

7. The method of claim 6, further comprising initializing said physical clock independently of setting said programmable register.

A

8. The method of claim 4, wherein said instruction is a STORE CLOCK EXTENDED instruction.

9. The method of claim 4, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

10. The method of claim 1, further comprising receiving said timing information from a physical clock of said computing environment.

11. The method of claim 10, further comprising initializing said physical clock to a predefined value using a set instruction.

12. The method of claim 11, further comprising obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

13. The method of claim 1, further comprising obtaining said selected information from a programmable register set by a set register instruction.

A3

14. (AMENDED) A method of generating sequence values usable within a computing environment, said method comprising:

Ar

A3

providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and

including as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

15. (AMENDED) The method of claim 14, further comprising providing as a further part of said monotonically increasing sequence value placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

Sub B2

16. (AMENDED) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and providing a timestamp, said representation comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to make the timestamp unique

A3 B2

across a plurality of operating system images on one or more processors of said computing environment.

17. CANCEL

Sub B3 A4

18. (AMENDED) The memory of claim 16, wherein said representation further comprises a placeholder component usable in ensuring that said timestamp is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

19. The memory of claim 16, wherein said representation further comprises a processor identifier component including processor information.

A5

20. (AMENDED) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and providing a monotonically increasing sequence value, said representation comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to ensure said monotonically increasing sequence value is unique across a

A

A3

plurality of operating system images on one or more processors of said computing environment.

21. (AMENDED) The memory of claim 20, wherein said representation further comprises:

a placeholder component usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

~~22. The memory of claim 21, wherein said representation further comprises a processor component including processor information.~~

Sub A7

A4

23. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

means for providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

means for including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

24. The system of claim 23, further comprising means for providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

25. The system of claim 23, further comprising means for providing as a further part of said sequence value a processor identifier.

26. The system of claim 23, wherein said means for providing and said means for including comprise an instruction.

27. The system of claim 26, wherein said means for providing comprises means for retrieving, by said instruction, said timing information from a physical clock, and wherein said means for including comprises means for retrieving, by said instruction, said selected information from a storage area.

28. The system of claim 27, wherein said storage area is a programmable register set by a set register instruction.

29. The system of claim 28, further comprising means for initializing said physical clock independently of setting said programmable register.

30. The system of claim 26, wherein said instruction is issued by a program of said computing environment

desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

31. The system of claim 23, further comprising means for receiving said timing information from a physical clock of said computing environment.

32. The system of claim 31, further comprising means for initializing said physical clock to a predefined value using a set instruction.

33. The system of claim 32, further comprising means for obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

34. The system of claim 23, further comprising means for obtaining said selected information from a programmable register set by a set register instruction.

---

35. (AMENDED) A system of generating sequence values usable within a computing environment, said system comprising:

B

means for providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and

A1

means for including as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

36. (AMENDED) The system of claim 35, further comprising means for providing as a further part of said monotonically increasing sequence value placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

Sub B5

37. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

said processor being further adapted to provide as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

A7

38. (AMENDED) A system of generating sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information;

said processor being further adapted to include as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment; and

said processor being further adapted to provide as another part of said monotonically increasing sequence value placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

Sub B6

39. (AMENDED) An article of manufacture, comprising:

at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising:

A7
B6

computer readable program code means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

computer readable program code means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

40. The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

41. The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a processor identifier.

42. The article of manufacture of claim 39, wherein said computer readable program code means for causing a computer to provide comprises computer readable program code means for causing a computer to retrieve, by an instruction, said timing information from a physical clock, and wherein

A

said computer readable program code means for causing a computer to include comprises computer readable program code means for causing a computer to retrieve, by said instruction, said selected information from a storage area.

43. The article of manufacture of claim 42, wherein said storage area is a programmable register set by a set register instruction.

44. The article of manufacture of claim 43, further comprising computer readable program code means for causing a computer to initialize said physical clock independently of setting said programmable register.

45. The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to receive said timing information from a physical clock of said computing environment.

46. The article of manufacture of claim 45, further comprising computer readable program code means for causing a computer to initialize said physical clock to a predefined value using a set instruction.

47. The article of manufacture of claim 46, further comprising computer readable program code means for causing a computer to obtain said selected information from a programmable register independent from said physical clock and said set instruction.

48. (AMENDED) At least one program storage device readable by a machine, tangibly embodying at least one

A6

program of instructions executable by the machine to perform a method of generating sequence values usable within a computing environment, said method comprising:

providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and

including as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

49. (AMENDED) The at least one program storage device of claim 48, wherein said method further comprises providing as a further part of said monotonically increasing sequence value placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

Remarks

Entry of the amendments, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. Claims 1-16 and 18-49 remain pending. Support for the amendments can be found throughout the specification (e.g., pages 28-32). Thus, no new matter is being added.

In accordance with 37 C.F.R. 1.121, a marked-up version of the amended specification and claims is provided on one or more pages separate from the amendment. These pages are appended at the end of the Response.

In the Office Action dated January 23, 2002, claims 1, 4-13, 16, 17, 23, 26-34, 37, 39, and 42-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Frey et al. (U.S. Patent No. 5,416,921); and claims 1, 3, 4, 16, 19, 23, 25, 26, 37, 39 and 41 are rejected under 35 U.S.C. 102(b) as being anticipated by Cwiakala et al. (U.S. Patent No. 5,257,379). Additionally, claims 2, 14, 15, 18, 20, 21, 24, 35, 36, 38, 40, 48 and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Frey and Sugiyama (U.S. Patent No. 5,933,625); and claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Frey and Sugiyama and further in view of Cwiakala. Applicants respectfully, but most strenuously, traverse these rejections for the reasons below.

In one aspect, applicants' invention is directed to generating timestamps that are unique across multiple operating system images. The timestamps are monotonically

increasing in value in that two different timestamps resulting from two instructions either on the same CPU or different CPUs correctly imply the sequence of execution of the two instructions.

As one example, applicants claim a method of generating unique sequence values usable within a computing environment (e.g., claim 1). The method includes, for instance, providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, the sequence value representing a timestamp; and including as another part of the sequence value, selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment. Thus, in one aspect of applicants' claimed invention, a timestamp is generated that is unique across a plurality of operating system images. This is very different from the teachings of any of the references, either alone or in combination.

For example, while Frey describes a technique for creating a unique value across multiple operating system images, this unique value is not and cannot represent a timestamp. In Frey, information is taken from a timestamp in order to make the unique value; however, the unique value in and of itself is no longer a timestamp. That is, the value generated in Frey is not a timestamp. There is no discussion at all in Frey of maintaining the generated value as an increasing sequence value, which is needed for a timestamp. Further, the new value does not have a well-defined anchor bit (i.e., a bit that represents the time value, such as a second, microsecond, etc.) needed for a

timestamp. Thus, the created value of Frey cannot represent a timestamp. Therefore, Frey does not describe, teach or suggest a timestamp that is unique across multiple operating system images, as claimed by applicants.

Similarly, Cwiakala also fails to describe, teach, or suggest a timestamp that is unique across operating system images. Again, while Cwiakala describes using information generated from a clock to create a token, the token is not usable as a timestamp. In Cwiakala, like Frey, the value is created by combining a number of different fields, one of which includes time information; however, the amalgamation of the different fields results in a value that is no longer usable as a timestamp. For example, there is no discussion in Cwiakala (or Frey) of ensuring that the created value is an increasing sequence value, as needed in a timestamp. There is no discussion in Cwiakala of creating a value that represents a time value, as claimed by applicants. Thus, Cwiakala also does not describe, teach or suggest a timestamp that is unique across multiple operating system images. Therefore, Cwiakala does not anticipate applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that independent claim 1, as well as independent claims 16, 23, 37 and 39 are patentable over Frey and Cwiakala. Additionally, the claims that depend from those independent claims are patentable for the same reasons as the independent claims, as well as for their own additional features.

For example, dependent claim 4 recites that the providing and the including elements of independent claim 1 are performed by an instruction. That is, in one aspect of applicants' claimed invention, the timestamp is generated by an instruction (e.g., a Store Clock Extended instruction). This is very different from the teachings of either Frey or Cwiakala, in which the value that is generated in those references is generated by programs combining various fields. While one of those fields is provided using an instruction (e.g., a Store Clock instruction), the other fields used to create the actual value and the combination of the fields are performed by programs and not instructions, as claimed by applicants. Thus, applicants' respectfully request withdrawal of the rejection of claim 4, as well as similar claims.

In addition to the above, it is respectfully submitted that independent claims 14, 20, 35, 38 and 48, which are rejected as being obvious in view of Frey and Sugiyama, are patentable over the teachings of Frey and Sugiyama, either alone or in combination. As one example, independent claim 14 recites a method of generating sequence values usable within a computing environment. The method includes, for instance, providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and including as another part of the monotonically increasing sequence value selected information usable in making the monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of the computing environment. Thus, in one aspect of applicants' claimed invention, a monotonically

increasing sequence value that is unique across a plurality of operating system images is being claimed.

As described above, Frey fails to describe, teach, or suggest monotonically increasing sequence values. Again, while Frey uses a portion of a timestamp to create a value, there is no discussion in Frey that the resulting value is to be an increasing sequence value. Since Frey does not describe, teach, or suggest that the value is to be an increasing sequence value, Frey does not teach or suggest applicants' claimed invention.

Further, Sugiyama does not teach or suggest applicants' claimed invention. While Sugiyama describes a timestamp, Sugiyama makes no mention whatsoever of uniqueness across different operating system images. Thus, Sugiyama does not teach or suggest applicants' claimed invention.

The combination of Frey and Sugiyama also does not teach or suggest applicants' claimed invention. In combination, the references teach, at most, the use of timestamps without regard to uniqueness across multiple operating system images; and the generation of values that are unique across operating system images, but are not monotonically increasing sequence values (or timestamps). There is no discussion, teaching or suggestion in the combination of generating monotonically increasing sequence values that are unique across multiple operating system images. That is, there is no teaching or suggestion in the references of how one would create an increasing sequence value that is unique across operating system images. This is missing from the teachings of the references. Thus, the

combination of Frey and Sugiyama fails to teach or suggest applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that independent claims 14, 20, 35, 38 and 48 are patentable over the combination of Frey and Sugiyama. Further, the dependent claims are patentable for the same reasons as the independent claims, as well as for their own additional features.

Based on the foregoing, applicants respectfully submit that all pending claims are patentable over the references, either alone or in combination.

Should the Examiner wish to discuss this case with applicants' attorney, please contact applicants' attorney at the below listed number.

Respectfully submitted,

Blanche E. Schiller
Blanche E. Schiller
Attorney for Applicants
Registration No. 35,670

Dated: June 24, 2002

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

Version with markings to show changes made

In the Specification:

On page 1, paragraph 2 of the Cross-Reference to Related Applications has been amended, as follows:

"METHOD, SYSTEM AND PROGRAM PRODUCTS FOR EMPLOYING EXPANDED PHYSICAL CLOCKS," by Elko et al., Serial No. [___________] 09/337,157, (Docket No. PO9-99-064), filed June 21, 1999.

In the Claims:

Claims 1, 14-16, 18, 20-21, 23, 35-39, 48 and 49 have been amended, as follows:

1. (AMENDED) A method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

14. (AMENDED) A method of generating sequence values usable within a computing environment, said method comprising:

providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and

including as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

[including as another part of said sequence value placeholder information usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.]

15. (AMENDED) The method of claim 14, further comprising providing as a further part of said monotonically increasing sequence value [selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.] placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

16. (AMENDED) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and <u>providing a timestamp, said representation</u> comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to [provide from said representation a sequence value that is] <u>make the timestamp</u> unique across a plurality of operating system images on one or more processors of said computing environment.

18. (AMENDED) The memory of claim 16, wherein said representation further comprises a placeholder component usable in ensuring that said [sequence value] <u>timestamp</u> is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

20. (AMENDED) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within [in] a computing environment and <u>providing a monotonically increasing sequence value, said representation</u> comprising:

a timing component comprising timing information including at least one of time-of-day information and date information[, said timing information being at least a part of a value resulting from said representation]; and

a programmable field component comprising selected information to ensure said monotonically increasing sequence value is unique across a plurality of operating system images on one or more processors of said computing environment.

[a placeholder component usable in ensuring that said value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.]

21. (AMENDED) The memory of claim 20, wherein said representation further comprises:

[a programmable field component comprising selected information to ensure said value is unique across a plurality of operating system images on one or more processors of said computing environment.]

a placeholder component usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

23. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

means for providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

means for including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

35. (AMENDED) A system of generating sequence values usable within a computing environment, said system comprising:

means for providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and

means for including as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment [including as another part of said sequence value placeholder information usable in ensuring that said sequence value is an increasing

sequence value, even when a physical clock used to provide said timing information wraps back to zero].

36. (AMENDED) The system of claim 35, further comprising means for providing as a further part of said monotonically increasing sequence value placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero [selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment].

37. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp; and

said processor being further adapted to provide as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

38. (AMENDED) A system of generating sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a <u>monotonically increasing</u> sequence value timing information comprising at least one of time-of-day information and date information; [and]

said processor being further adapted to <u>include as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment; and</u>

said processor being further adapted to provide as another part of said <u>monotonically increasing</u> sequence value placeholder information usable in ensuring that said <u>monotonically increasing</u> sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

39. (AMENDED) An article of manufacture, comprising:

at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising:

computer readable program code means for causing a computer to provide as one part of a sequence value timing information comprising at

least one of time-of-day information and date information, said sequence value representing a timestamp; and

computer readable program code means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

48. (AMENDED) At least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method of generating sequence values usable within a computing environment, said method comprising:

providing as one part of a monotonically increasing sequence value timing information comprising at least one of time-of-day information and date information; and

including as another part of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment [including as another part of said sequence value placeholder information usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero].

49. (AMENDED) The at least one program storage device of claim 48, wherein said method further comprises providing as a further part of said <u>monotonically increasing</u> sequence value <u>placeholder information usable in ensuring that said monotonically increasing sequence value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero</u> [selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment].

Claim 17 has been canceled.

K

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D C 20231
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/337,158	06/21/1999	DAVID ARLEN ELKO	PO9-99-007	8602

7590 09/26/2002

BLANCHE E SCHILLER ESQ
HESLIN & ROTHENBERG P C
5 COLUMBIA CIRCLE
ALBANY, NY 12203

EXAMINER	
PARK, ILWOO	
ART UNIT	PAPER NUMBER
2182	7

DATE MAILED: 09/26/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Office Action Summary	Application No. 09/337,158	Applicant(s) ELKO ET AL.
	Examiner Ilwoo Park	Art Unit 2182

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 July 2002*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-16 and 18-49* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-16 and 18-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

DETAILED ACTION

1. Applicant's amendment filed on 7/1/2002 in response to Examiner's Office Action has been reviewed. Claims 17 is canceled and claims 1, 14-16, 18, 20, 21, 23, 35-39, 48, and 49 are amended. The following rejections now apply.

2. Claims 1-16 and 18-49 are presented for examination.

3. Frey et al, Cwiakala et al, and Sugiyama were cited as prior art in the last office action.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 1-13, 16, 18, 19, 23-34, 37, and 39-47 are rejected under 35 U.S.C. 112, first paragraph, as containing subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The limitation "timestamp" is not disclosed and defined in the specification.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 1, 4-13, 16, 23, 26-34, 37, 39, and 42-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Frey et al., US patent No. 5,416,921.

As to claims 1, 16, 23, 37, and 39, Frey et al teach [col. 20, line 60-col. 21, line 17] a method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part [time value from system clock 60] of a sequence value [col. 20, lines 54-66] timing information comprising at least one of time-of-day information and date information, said sequence value representing a timestamp [col. 12, lines 55-68]; and

including as another part [system identifier] of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images [col. 2, lines 23-37] on one or more processors of said computing environment.

8. As to claims 4, 8, and 26, Frey et al teach said providing and said including are performed by an instruction [col. 20, lines 60-68].

9. As to claims 5, 10, 13, 27, 31, 34, 42, 43, and 45, Frey et al teach retrieving by an instruction said timing information from a physical clock [system clock 60] and retrieving said selected information from a storage [col. 20, line 60-col. 21, line 17].

10. As to claims 6 and 28, Frey et al teach said storage area is a programmable register [col. 20, line 60-col. 21, line 17] by a set register instruction [inherent: col. 2, lines 23-26].

11. As to claims 7, 29, and 44, Frey et al teach initializing said physical clock independently of setting said programmable register [col. 12, lines 55-68].

12. As to claims 9 and 30, Frey et al teach initializing said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value [col. 20, line 60-col. 21, line 17] .

13. As to claims 11, 32, and 46, Frey et al teach initializing said physical clock to a predetermined value using a set instruction [inherent: col. 2, lines 23-26; col. 12, lines 55-68; col. 20, line 60-col. 21, line 17].

14. As to claims 12, 33, and 47, Frey et al teach obtaining said selected information from a programmable register independent from said physical clock and said set instruction [col. 20, line 60-col. 21, line 17].

15. Claims 1, 3, 4, 16, 19, 23, 25, 26, 37, 39, and 41 are rejected under 35 U.S.C. 102(b) as being anticipated by Cwiakala et al., US patent No. 5,257,379.

As to claims 1, 16, 23, 37, and 39, Cwiakala et al teach [col. 17, line 25-col. 18, line 17] a method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part [time-of-day clock 1101E] of a sequence value timing information comprising at least one [inherent] of time-of-day information and data information, said sequence value representing a timestamp; and

including as another part [operating system configuration identifier 1101F] of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

As to claims 1, 16, 23, 37, and 39, Frey et al teach [col. 20, line 60-col. 21, line 17] a method of generating unique sequence values usable within a computing environment, said method comprising:

16. As to claims 3, 19, 25, and 41, Cwiakala et al teach providing as a further part of said sequence value a processor identifier [col. 17, lines 48-50].

17. As to claims 4 and 26, Cwiakala et al teach said providing and said including are performed by an instruction [col. 31, lines 34-42].

18. As to claims 14, 20, 35, and 48, Frey et al teach a method of generating sequence values usable within a computing environment, said method comprising:

providing as one part [time value from system clock 60] of a monotonically increasing sequence value [col. 20, lines 54-66] timing information comprising at least one [inherent] of time-of-day information and date information; and

including as another part [system identifier] of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images [col. 2, lines 23-37] on one or more processors of said computing environment.

19. (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371© of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) do not apply to the examination of this application as the application being examined was not (1) filed on or after November 29, 2000, or (2) voluntarily published under 35 U.S.C. 122(b). Therefore, this application is examined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

20. Claims 14, 20, 35, and 48 are rejected under 35 U.S.C. 102(e) as being anticipated by Lyle et al., US patent No. 6,363,389.

As to claims 14, 20, 35, and 48, Lyle et al teach a method of generating sequence values [unique row identification numbers] usable within a computing environment, said method comprising:

providing as one part [computer-dependent timestamp: fig. 7 and] of a monotonically increasing sequence value [fig. 9] timing information comprising at least one of time-of-day information and date information; and

including as another part [computer-dependent information: fig. 7 and col. 8, lines 1-32] of said monotonically increasing sequence value selected information usable in making said monotonically increasing sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

Claim Rejections - 35 USC § 103

21. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

22. Claims 2, 15, 18, 21, 24, 36, 40, and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Frey et al. as applied to claims 1, 14, 16, 20, 23, 35, 39, and 48 above, and further in view of Sugiyama, US patent No. 5,933,625.

As claims 2, 15, 18, 21, 24, 36, 40, and 49, Sugiyama teach a method of generating unique sequence values usable within a computing environment [col. 3, lines 26-50], said method comprising providing as a further part of sequence value a placeholder value [col. 9, line 63-col. 10, line 3] usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide timing information of said sequence value wraps back to zero.

Therefore, it would have been obvious to one of ordinary skill in the art to combine the teachings of Frey et al and Sugiyama at the time of the invention because they both teach a method of generating unique sequence values using a timing information from a physical clock and the Sugiyama's teaching of including as a part of the sequence value the placeholder value would further increase uniqueness of the Frey et al's sequence values [Frey et al: col. 5, line 62-col. 6, line 18].

23. Claim 38 is rejected under 35 U.S.C. 103(a) as being unpatentable over Frey et al., US patent No. 5,416,921 and Sugiyama, US patent No. 5,933,625.

As to claim 38, Frey et al. and Sugiyama teach the claimed invention [see above].

24. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Frey et al., US patent No. 5,416,921 and Sugiyama, US patent No. 5,933,625 as applied to claim 21 above, and further in view of Cwiakala et al., US patent No. 5,257,379.

As to claim 22, Cwiakala et al teach providing as a further part of said sequence value a processor information [col. 17, lines 48-50].

Therefore, it would have been obvious to one of ordinary skill in the art to combine the teachings of Frey et al and Sugiyama at the time of the invention to include a processor identifier in the sequence value in order to further increase uniqueness.

Response to Arguments

25. Applicant's arguments filed 7/1/02 have been fully considered but they are not persuasive. In the remarks, applicants argued in substance that there is no teaching or suggestion that a) Frey et al's unique value is not and cannot represent a timestamp and b) Frey et al fail to teach monotonically increasing sequence values.

As to point a), first of all, the arguments contain the limitation "timestamp" which is not disclosed and defined in the specification. And Frey et al's each of unique values is generated by

combining a time value with a system identifier [col. 20, line 60-col. 21, line 5] in the IBM's MVS system; the ever increasing time value is for writing timestamps [col. 12, lines 55-59]. Thus, Frey's each of unique values is a time stamped value.

As to point b), Frey et al's each of unique values generated is a monotonically increasing sequence value because the value is monotonically increased by the ever increasing time value [col. 20, line 60-col. 21, line 5].

Conclusion

26. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

17. Any inquiry concerning this communication should be directed to Ilwoo Park, whose telephone number is (703) 308-7811 or via e-mail, ***ilwoo.park@uspto.gov***. The Examiner can normally be reached Monday through Friday from 9:00 AM to 5:30 PM.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Jeffrey A. Gaffin, can be reached at (703) 308-3301.

Any inquiry of a general nature of relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 746-7239 (for formal communications intended for entry),

(703) 746-7238 (for after-final communications),

or:

(703) 746-7240 (for informal or draft communications, please label "PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA., 4th Floor (Receptionist).

JEFFREY GAFFIN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Ilwoo Park

Ilwoo Park

September 24, 2002

Notice of References Cited	Application/Control No. 09/337,158	Applicant(s)/Patent Under Reexamination ELKO ET AL.	
	Examiner Ilwoo Park	Art Unit 2182	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,363,389	03-2002	Lyle et al.	707/1
*	B	US-5,416,921	05-1995	Frey et al.	714/11
*	C	US-5,257,379	10-1993	Cwlakala et al.	710/7
*	D	US-5,933,625	08-1999	Sugiyama, Akira	713/503
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

#8
Gary
2-13-03

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) (Large Entity)	Docket No. PO9-99-007

In Re Application Of: **Elko et al.**

Serial No.	Filing Date	Examiner	Group Art Unit
09/337,158	06/21/99	Ilwoo Park	2182

Invention: **METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES**

RECEIVED
FEB 1 2 2003
Technology Center 2100

OIPE
FEB 1 0 2003
PATENT & TRADEMARK OFFICE

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response to the Office Action of September 26, 2002 (*Date*) above-identified application.

The requested extension is as follows (check time period desired):

☐ One month ☒ Two months ☐ Three months ☐ Four months ☐ Five months

from: **December 26, 2002** (*Date*) until: **February 26, 2003** (*Date*)

The fee for the extension of time is **$410** and is to be paid as follows:

☒ A check in the amount of the fee is enclosed.

☐ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account No.
A duplicate copy of this sheet is enclosed.

☒ If an additional extension of time is required, please consider this a petition therefor and charge any additional fees which may be required to Deposit Account No. **08-1935**
A duplicate copy of this sheet is enclosed.

Blanche E. Schiller
Signature

Dated: **February 10, 2003**

Blanche E. Schiller, Esq.
Reg. No. 35,670
HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

02/12/2003 MBERHE 00000122 090463 09337158
01 FC:1252 410.00 OP

cc:

Certificate of Express Mail

I hereby certify that this correspondence is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on February 10, 2003 and addressed to BOX CPA, Commissioner for Patents, Washington D.C. 20231, Express Mail Label No.: EV 245035549 US

Sharon Kaminski
Sharon Kaminski

P12LARGE/REV06

:ss Mail Label No. EV 24593! US 2-11-3

CYM/ 2182

CONTINUED PROSECUTION APPLICATION (CPA) REQUEST TRANSMITTAL (Large Entity)

Submit an original, and a duplicate for fee processing.

(Only for Continuation or Divisional Applications Under 37 CFR 1.53(d))

Docket No. PO9-99-007

☐ DUPLICATE (Check box if applicable)

First Named Inventor	Examiner	Group/Art Unit
David A. Elko	Ilwoo Park	2182

OIPE FEB 1 0 2003 PATENT & TRADEMARK OFFICE

RECEIVED FEB 1 2 2003 Technology Center 2100

Address to:
Assistant Commissioner for Patents
Box CPA
Washington, D.C. 20231

This is a request for filing a ☒ continuation, or ☐ divisional application under 37 CFR 1.53(d), (continued prosecution application (CPA)) of prior application number 09/337,158 filed on June 21, 1999 and entitled:

METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

1. ☐ Enter the unentered amendment previously filed on ______________ under 37 CFR 1.116 in the prior nonprovisional application.
2. ☒ A preliminary amendment is enclosed.
3. ☐ This application is being filed by fewer than all the inventors named in the prior application, 37 CFR 1.53(d)(4).
   a. ☐ *DELETE* the following inventor(s) named in the prior nonprovisional application:

02/12/2003 MBERHE 00000122 090463 09337158
02 FC:1006 750.00 CH
03 FC:1202 324.00 CH
04 FC:1201 168.00 CH

   b. ☐ The inventor(s) to be deleted are set forth on a separate sheet attached hereto.
4. ☐ A new power of attorney or authorization of agent is enclosed.
5. ☐ An Information Disclosure Statement (IDS) is enclosed:
   a. ☐ PTO-1449
   b. ☐ Copies of IDS Citations
6. ☐ The fee for this application is calculated as follows:

CLAIMS AS FILED

For	#Filed	#Allowed	#Extra		Rate	Fee
Total Claims	38	- 20 =	18	x	$18.00	$324.00
Indep. Claims	5	- 3 =	2	x	$84.00	$168.00
Multiple Dependent Claims (check if applicable) ☐						$0.00
					BASIC FEE	$750.00
					TOTAL FILING FEE	$1,242.00

CONTINUED PROSECUTION APPLICATION (CPA) REQUEST TRANSMITTAL (Large Entity)
(Only for Continuation or Divisional Applications Under 37 CFR 1.53(d))

7. ☒ The Commissioner is hereby authorized to credit overpayments or charge the following fees to Deposit Account No. **09-0463 (IBM)**

   ☒ fees required under 37 C.F.R. 1.16.
   ☒ fees required under 37 C.F.R. 1.17.
   ☐ fees required under 37 C.F.R. 1.18.

8. ☐ A check in the amount of ________________ is enclosed.

9. ☒ Also enclosed:
   Certificate of Mailing by Express Mail
   Petition for Extension of Time
   Check for $410 (Extension Fee)
   One (1) Acknowledgment Postcard

10. ☐ The prior application's correspondence address will carry over to this CPA UNLESS a new correspondence address is provided below:

Expr

CONTINUED PROSECUTION APPLICATION (CPA) REQUEST TRANSMITTAL (Large Entity)
(Only for Continuation or Divisional Applications Under 37 CFR 1.53(d))

NOTES

Submit an original, and a duplicate for fee processing.

FILING QUALIFICATIONS: The prior application must be a nonprovisional application that is either: (1) complete as defined by 37 C.F.R. 1.51(b), or (2) the national stage of an international application in compliance with 35 U.S.C. 371. A Notice will be placed on a patent issuing from a CPA, except for reissues and designs, to the effect that the patent issued on a CPA and is subject to the twenty-year patent term provisions of 35 USC 154(a)(2). Therefore, the prior application of a CPA may have been filed before, on or after June 8, 1995.

C-I-P NOT PERMITTED: A continuation-in-part application cannot be filed as a CPA under 37 C.F.R. 1.53(d), but must be filed under 37 C.F.R. 1.53(b).

EXPRESS ABANDONMENT OF PRIOR APPLICATION: The filing of this CPA is a request to expressly abandon the prior application as of the filing date of the request for a CPA. 37 C.F.R. 1.53(b) must be used to file a continuation, divisional or continuation-in-part of an application that is not to be abandoned.

ACCESS TO PRIOR APPLICATION: The filing of this CPA will be construed to include a waiver of confidentiality by the Applicant under 35 U.S.C. 122 to the extent that any member of the public who is entitled under the provisions of 37 C.F.R. 1.14 to access to, copies of, or information concerning, the prior application may be given similar access to, copies of, or similar information concerning, the other application or application in the file jacket.

35 U.S.C. 120 STATEMENT: In a CPA, no reference to the prior application is needed in the first sentence of the specification and none should be submitted. If a sentence referencing the prior application is submitted, it will not be entered. A request for a CPA is the specific reference required by 35 U.S.C. 120 and to every application assigned the application number identified in such request; 37 C.F.R. 1.78(a).

Dated: February 10, 2003

Blanche E. Schiller
Signature

Blanche E. Schiller
Typed or printed name

35,670
Registration Number (if applicable)

☐ Inventor(s)

☐ Assignee of complete interest

☒ Attorney or agent of record

cc:

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

Applicant: Elko et al. Confirmation No.: 8602

Serial No.: 09/337,158 Group Art Unit: 2182

Filed: 06/21/99 Examiner: Ilwoo Park

Title: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

"EXPRESS MAIL" MAILING LABEL NO. EV 245035549 US

Date of Deposit: February 10, 2003

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and addressed to:

Box CPA
Commissioner for Patents
Washington, D.C. 20231

Sharon Kaminski
(Typed or printed name of person mailing paper or fee)

Sharon Kaminski
(Signature of person mailing paper or fee)

Enclosures:

*Continued Prosecution Application Request Transmittal (3 pages - in duplicate)
*Preliminary Amendment (18 pages)
*Petition for Extension of Time (1 page - in duplicate)
*Check for $410 (Extension Fee)
*Acknowledgment Postcard

#10/B
Pair
2-13-03

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Elko et al.	Confirmation No.: 8602
Serial No.:	09/337,158	Group Art Unit: 2182
Filed:	06/21/99	Examiner: Ilwoo Park

Title: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

Certificate of Express Mail

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on February 10, 2003 and addressed to Box CPA, Commissioner for Patents, Washington, D.C. 20231, Express Mail Label No.: EV 245035549 US.

Sharon Kaminski
Sharon Kaminski

Date of Signature: February 10, 2003

Box CPA
Commissioner for Patents
Washington, D.C. 20231

Preliminary Amendment

Dear Sir:

Prior to examination of the above-referenced application, please amend the application as follows:

Amendments

Please amend the above-identified application, as follows:

In the Claims:

Kindly cancel claims 14-15, 20-22, 35-36, 38, and 48-49, without prejudice, and amend claims 1, 16, 18, 23, 37 and 39, as follows. All claims are reproduced below for the Examiner's convenience.

B1

1. (AMENDED) A method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

2. The method of claim 1, further comprising providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

3. The method of claim 1, further comprising providing as a further part of said sequence value a processor identifier.

4. The method of claim 1, wherein said providing and said including are performed by an instruction.

5. The method of claim 4, wherein said providing comprises retrieving, by said instruction, said timing information from a physical clock, and wherein said including comprises retrieving, by said instruction, said selected information from a storage area.

6. The method of claim 5, wherein said storage area is a programmable register set by a set register instruction.

7. The method of claim 6, further comprising initializing said physical clock independently of setting said programmable register.

8. The method of claim 4, wherein said instruction is a STORE CLOCK EXTENDED instruction.

505D1

9. The method of claim 4, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

10. The method of claim 1, further comprising receiving said timing information from a physical clock of said computing environment.

11. The method of claim 10, further comprising initializing said physical clock to a predefined value using a set instruction.

12. The method of claim 11, further comprising obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

13. The method of claim 1, further comprising obtaining said selected information from a programmable register set by a set register instruction.

14. CANCELED

15. CANCELED

B2

16. (AMENDED) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and providing a value usable as a current time of day value, said representation comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to make the value unique across a plurality of operating system images on one or more processors of said computing environment.

17. PREVIOUSLY CANCELED

B3

18. (AMENDED) The memory of claim 16, wherein said representation further comprises a placeholder component usable in ensuring that said value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

19. The memory of claim 16, wherein said representation further comprises a processor identifier component including processor information.

20. CANCELED

21. CANCELED

22. CANCELED

23. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

B4

means for providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

means for including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

24. The system of claim 23, further comprising means for providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

SubD1

25. The system of claim 23, further comprising means for providing as a further part of said sequence value a processor identifier.

26. The system of claim 23, wherein said means for providing and said means for including comprise an instruction.

27. The system of claim 26, wherein said means for providing comprises means for retrieving, by said instruction, said timing information from a physical clock, and wherein said means for including comprises means for retrieving, by said instruction, said selected information from a storage area.

28. The system of claim 27, wherein said storage area is a programmable register set by a set register instruction.

29. The system of claim 28, further comprising means for initializing said physical clock independently of setting said programmable register.

30. The system of claim 26, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

31. The system of claim 23, further comprising means for receiving said timing information from a physical clock of said computing environment.

32. The system of claim 31, further comprising means for initializing said physical clock to a predefined value using a set instruction.

33. The system of claim 32, further comprising means for obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

Sub D1

34. The system of claim 23, further comprising means for obtaining said selected information from a programmable register set by a set register instruction.

35. CANCELED

36. CANCELED

B5

37. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

B5a end

said processor being further adapted to provide as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

38. CANCELED

39. (AMENDED) An article of manufacture, comprising:

B6

at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising:

computer readable program code means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

505 D1

computer readable program code means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

40. The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

41. The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a processor identifier.

42. The article of manufacture of claim 39, wherein said computer readable program code means for causing a computer to provide comprises computer readable program code means for causing a computer to retrieve, by an instruction, said timing information from a physical clock, and wherein said computer readable program code means for causing a computer to include comprises computer readable program code means for causing a computer to retrieve, by said instruction, said selected information from a storage area.

43. The article of manufacture of claim 42, wherein said storage area is a programmable register set by a set register instruction.

44. The article of manufacture of claim 43, further comprising computer readable program code means for causing a computer to initialize said physical clock independently of setting said programmable register.

45. The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to receive said timing information from a physical clock of said computing environment.

46. The article of manufacture of claim 45, further comprising computer readable program code means for causing a computer to initialize said physical clock to a predefined value using a set instruction.

47. The article of manufacture of claim 46, further comprising computer readable program code means for causing a computer to obtain said selected information from a programmable register independent from said physical clock and said set instruction.

48. CANCELED

49. CANCELED

Sub D1

Remarks

Entry of the amendments, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. After entry of the amendments, claims 1-13, 16, 18-19, 23-34, 37 and 39-47 are pending.

In accordance with 37 C.F.R. 1.121(c)(1)(ii), a marked-up version of the amended claims is provided on one or more pages separate from the amendment. These pages are appended at the end of the Response.

In the Final Office Action dated September 26, 2002, claims 1-13, 16, 18, 19, 23-34, 37 and 39-47 are rejected under 35 U.S.C. §112, first paragraph, as containing subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. In particular, it is stated, "The limitation 'timestamp' is not disclosed and defined in the specification." Although applicants believe that "timestamp" is supported in the application, applicants have amended the claims to recite that the sequence value is usable as a current time of day value, in order to further prosecution of this application. Applicants respectfully submit that the amended language is supported throughout the application (e.g., page 9, lines 1-3; page 11, lines 1-18, etc.), and therefore, no new matter is being added. Based on the foregoing, applicants respectfully submit that the claims satisfy §112.

In addition to the above, claims 1, 4-13, 16, 23, 26-34, 37, 39, and 42-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Frey et al. (U.S. Patent No. 5,416,921); claims 1, 3, 4, 16, 19, 23, 25, 26, 37, 39, and 41 are rejected under 35 U.S.C. 102(b) as being anticipated by Cwiakala et al. (U.S. Patent No. 5,257,379); claims 14, 20, 35, and 48 are rejected under 35 U.S.C. 102(e) as being anticipated by Lyle et al. (U.S. Patent No. 6,363,389); and claims 2, 15, 18, 21, 24, 36, 38, 40, and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Frey et al. in view of Sugiyama (U.S. Patent No. 5,933,625); and claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Frey and Sugiyama and further in view of Cwiakala et

al. (U.S. Patent No. 5,257,379). Applicants respectfully, but most strenuously, traverse these rejections for the reasons herein.

In one aspect, applicants' invention is directed to generating a sequence value that is unique across multiple operating system images and is usable as a current time of day value. That is, the time of day value reflects the time at which time is requested. The sequence value is, for instance, monotonically increasing in value in that two different values resulting from two instructions either on the same CPU or different CPU's correctly imply the sequence of execution of the two instructions.

As one example, applicants' claim a method of generating unique sequence values usable within a computing environment (e.g., claim 1). The method includes, for instance, providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, in which the sequence value is usable as a current time of day value; and including as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment. Thus, in one aspect of applicants' claimed invention, a sequence value is generated that is unique across a plurality of operating system images and is usable as a current time of day value. That is, the resulting sequence value still has the property of being able to be used as a time of day value. This is very different from the teachings of any of the references, either alone or in combination.

For example, while Frey describes a technique for creating a unique value across multiple operating system images, this unique value is not and cannot represent a value that is usable as a current time of day value. In Frey, information is taken from a timestamp in order to make the unique value; however, the unique value, once created, is no longer, in and of itself, a timestamp. That is, the value generated in Frey, which has time information embedded therein, is not a value that is usable as a current time of day value. This is because in Frey there is no requirement that the created values represent the current time of day and that the values be usable as time of day values. The only requirement is that the values be unique. Thus, as one example, the values can

be created at any time and then stored for later use. Further, they can be used in any order, since the time information is only used to ensure uniqueness and not a particular time of day.

Further, since the values in Frey do not necessarily represent current time, a processor, as one example, cannot use the values generated in Frey for timing purposes. In particular, as one example, in Frey, the high order six bytes (bits 0-47) are used in the authority parameter, which provides a resolution of the timestamp field as 16 microseconds. This equates to 16 thousand instructions on a gigahertz processor, which is not granular enough for programmatic timing uses. Further, users of Frey's values do not even have access to the fields because they are privileged. In contrast, applicants' sequence value is usable as a current time of day value. Applicants' value has, for instance, a particular format with a defined anchor (e.g., bit 59=1 microsecond), which allows for programmatic computations of time and time intervals. The value has, as one example, a precision that matches the precision of the physical clock and meets the architecture requirements of a resolution equal to around 10 instruction execution times. Thus, as shown, applicants' value is usable as a current time of day, while Frey's is not. Frey's value simply does not meet the architecture requirements for a time of day value, and thus, cannot be used as a current time of day value.

Support for the rejection is indicated in Frey at Col. 12, lines 55-59. Col. 12 describes an external time reference that provides time information. However, once that information is embedded in a parameter, that information is no longer usable as a current time of day value, as described above. Thus, although a value that is unique across a plurality of operating system images is provided and that value may include some information extracted from an external time reference, the resulting value is no longer usable as a current time of day value. There is no teaching or suggestion in Frey that the resulting value still represents a time value and is usable as such. This is simply missing from Frey.

Similarly, in Col. 20, lines 60-68, Frey indicates that the system clock produces an ever increasing time value. However, that system clock, when combined with other information, is again no longer usable as a time value. Instead, it is merely used as a subsystem authority parameter value. There is no teaching or suggestion in Frey that the authority parameter value is

usable as a current time of day value. Based on the foregoing, Frey does not describe, teach or suggest a value that is unique across multiple operating system images and can be used as a current time of day value, as claimed by applicants.

Similarly, Cwiakala also fails to describe, teach, or suggest a value that is unique across operating system images and is usable as a current time of day value. Again, while Cwiakala describes using information generated from a clock to create a token, the token is not usable as a current time of day value. In Cwiakala, like Frey, the value is created by combining a number of different fields, one of which includes time information; however, the amalgamation of the different fields results in a value that is no longer usable as a time value. For example, there is no discussion in Cwiakala (or Frey) of ensuring that the created value is an increasing sequence value, as needed in a time of day value. Again, while the clock itself produces increasing sequence values, once that clock information is combined with other information, there is no requirement in Cwiakala (or in Frey) of maintaining the increasing sequence value in the resulting value, so that it can be used for a current time of day value. Thus, there is no discussion in Cwiakala of creating a value that represents a time value, as claimed by applicants. Therefore, Cwiakala does not anticipate applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that independent claim 1, as well as independent claim 16, 23, 37 and 39, are patentable over Frey and Cwiakala. Additionally, the claims that depend from those independent claims are patentable for the same reasons as the independent claims, as well as for their own additional features.

For example, dependent claim 4 recites that the providing and the including elements of independent claim 1 are performed by an instruction. That is, in one aspect of applicants' claimed invention, the value is generated by an instruction (e.g., a Store Clock Extended instruction). This is very different from the teachings of either Frey or Cwiakala, in which the value that is generated in those references is generated by privileged programs combining various fields. While one of those fields is provided using an instruction (e.g., a Store Clock Instruction), the other fields used to create the actual value and the combination of the fields are performed by privileged programs and not instructions, as claimed by applicants. Thus,

applicants respectfully submit that claim 4, as well as similar claims, are patentable over Frey and Cwiakala.

In addition to the above, Sugiyama does not overcome the deficiencies of Frey and/or Cwiakala. While Sugiyama describes a timestamp, Sugiyama makes no mention whatsoever of uniqueness across different operating system images. Thus, Sugiyama does no teach or suggest applicants' claimed invention.

For example, the combination of Frey and Sugiyama teaches, at most, the use of timestamps without regard to uniqueness across multiple operating system images; and the generation of values that are unique across operating system images, but are not monotonically increasing sequence values (or values that are usable as current time of day values). There is no discussion, teaching or suggestion in the combination of generating unique sequence values that are usable as current time of day values. That is, there is no teaching or suggestion in the references of how one would create a sequence value that is unique across operating system images and is still usable as a current time of day value. This is missing from the teachings of the references. Thus, the combination of Frey and Sugiyama fails to teach or suggest applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that all pending claims are patentable over the references, either alone or in combination.

Should the Examiner wish to discuss this case with applicants' attorney, please contact applicants' attorney at the below listed number.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No. 35,670

Dated: February 10, 2003

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

Version with markings to show changes made

In the Claims:

Claims 1, 16, 18, 23, 37 and 39 have been amended, as follows:

1. (AMENDED) A method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value [representing a timestamp] being usable as a current time of day value; and

including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

16. (AMENDED) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and providing a [timestamp] value usable as a current time of day value, said representation comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to make the [timestamp] value unique across a plurality of operating system images on one or more processors of said computing environment.

18. (AMENDED) The memory of claim 16, wherein said representation further comprises a placeholder component usable in ensuring that said [timestamp] value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

23. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

means for providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value [representing a timestamp] being usable as a current time of day value; and

means for including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

37. (AMENDED) A system of generating unique sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value [representing a timestamp] being usable as a current time of day value; and

said processor being further adapted to provide as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

39. (AMENDED) An article of manufacture, comprising:

at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable

within a computing environment, the computer readable program code means in said article of manufacture comprising:

computer readable program code means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value [representing a timestamp] being usable as a current time of day value; and

computer readable program code means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

Claims 14-15, 20-22, 35-36, 38, and 48-49 have been canceled.

Sk

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D C 20231
www uspto gov

APPLICATION NO	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/337,158	06/21/1999	DAVID ARLEN ELKO	PO9-99-007	8602

7590 03/19/2003

BLANCHE E SCHILLER ESQ
HESLIN & ROTHENBERG P C
5 COLUMBIA CIRCLE
ALBANY, NY 12203

EXAMINER	
PARK, ILWOO	

ART UNIT	PAPER NUMBER
2182	11

DATE MAILED: 03/19/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Sk

Office Action Summary	Application No.	Applicant(s)
	09/337,158	ELKO ET AL.
	Examiner	Art Unit
	Ilwoo Park	2182

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 February 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-13,16,18,19,23-34,37 and 39-47* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-13,16,18,19,23-34,37 and 39-47* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. The request filed on 2/10/2003 for a Continued Prosecution Application (CPA) under 37 CFR 1.53(d) based on parent Application No. 09/337,158 is acceptable and a CPA has been established. An action on the CPA follows.

2. Applicant's amendment filed on 2/10/2003 in response to Examiner's Office Action has been reviewed. Claims 14-15, 17, 20-22, 35-36, 38, and 48-49 are canceled and claims 1, 16, 18, 23, 37, and 39 are amended. The following rejections now apply.

3. Claims 1-13, 16, 18-19, 23-34, 37, and 39-47 are presented for examination.

4. Frey et al, Cwiakala et al, Lyle et al, and Sugiyama were cited as prior art in the last office action.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1, 3-13, 16, 19, 23, 25-34, 37, 39, and 41-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Wanish, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pages 3819-3820, January, 1981.

As to claims 1, 16, 23, 37, and 39, Wanish teaches a method of generating unique sequence values [unique host identifiers] usable within a computing environment, said method comprising:

providing as one part [time of day clock] of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable [3rd paragraph in page 3820] as a current time of day value; and

including as another part [CPU address] of said sequence value selected information usable in making said sequence value unique across [2nd paragraph in page 3819] a plurality of operating system images on one or more processors of said computing environment [MV environments].

7. As to claims 3, 19, 25, and 41, Wanish teaches providing as a further part of said sequence value a processor identifier [CPU identification number].

8. As to claims 4, 8, and 26, Wanish teaches said providing and said including are performed by an instruction [STAP and STCK instruction].

9. As to claims 5, 10, 13, 27, 31, 34, 42, 43, and 45, Wanish teaches retrieving by an instruction said timing information from a physical clock and retrieving said selected information from a storage [see 3rd paragraph and the table for the storage format in page 3819].

10. As to claims 6 and 28, Wanish teaches said storage area is a programmable register by a set register instruction [see 3rd paragraph and the table for the storage format in page 3819].

11. As to claims 7, 29, and 44, Wanish teaches initializing said physical clock independently of setting said programmable register [page 3819].

12. As to claims 9 and 30, Wanish teaches initializing said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value [see 3rd paragraph in page 3819].

13. As to claims 11, 32, and 46, Wanish teaches initializing said physical clock to a predetermined value using a set instruction [see 3rd paragraph in page 3819].

14. As to claims 12, 33, and 47, Wanish teaches obtaining said selected information from a programmable register independent from said physical clock and said set instruction [page 3819].

Claim Rejections - 35 USC § 103

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

16. Claims 2, 18, 24, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wanish as applied to claims 1, 16, 23, and 39 above, and further in view of Sugiyama, US patent No. 5,933,625.

As claims 2, 18, 24, and 40, Sugiyama teach a method of generating unique sequence values usable within a computing environment [col. 3, lines 26-50], said method comprising

providing as a further part of sequence value a placeholder value [col. 9, line 63-col. 10, line 3] usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide timing information of said sequence value wraps back to zero.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Wanish and Sugiyama because they both teach a method of generating unique sequence values using a timing information from a physical clock and the Sugiyama's teaching of including as a part of the sequence value the placeholder value would further increase uniqueness of the Wanish's sequence values.

Response to Arguments

17. Applicant's arguments with respect to claims 1, 16, 23, 37, and 39 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

17. Any inquiry concerning this communication should be directed to Ilwoo Park, whose telephone number is (703) 308-7811 or via e-mail, ***ilwoo.park@uspto.gov***. The Examiner can normally be reached Monday through Friday from 9:00 AM to 5:30 PM.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Jeffrey A. Gaffin, can be reached at (703) 308-3301.

Any inquiry of a general nature of relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 746-7239 (for formal communications intended for entry),

(703) 746-7238 (for after-final communications),

or:

(703) 746-7240 (for informal or draft communications, please label "PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA., 4th Floor (Receptionist).

Ilwoo Park

Ilwoo Park

March 17, 2003

Notice of References Cited	Application/Control No. 09/337,158	Applicant(s)/Patent Under Reexamination ELKO ET AL.	
	Examiner Ilwoo Park	Art Unit 2182	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	JS-5,933,625	08-1999	Sugiyama, Akira	713/503
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	P. J. Wanish, January 1981, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pp 3819-3820
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

2182 #12 [illegible] 7/23/03

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) (Large Entity)	Docket No. PO9-99-007

In Re Application Of: **Elko et al.**

Serial No.	Filing Date	Examiner	Group Art Unit
09/337,158	06/21/99	Ilwoo Park	2182

Invention: **METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES**

OIPE JUL 1 4 2003 PATENT & TRADEMARK OFFICE

RECEIVED JUL 2 2 2003 Technology Center 2100

TO THE COMMISSIONER FOR PATENTS:

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response to the Office Action of March 19, 2003 (*Date*) above-identified application.

The requested extension is as follows (check time period desired):

☒ One month ☐ Two months ☐ Three months ☐ Four months ☐ Five months

from: **June 19, 2003** (*Date*) until: **July 19, 2003** (*Date*)

The fee for the extension of time is **$110** and is to be paid as follows:

☒ A check in the amount of the fee is enclosed.

☒ The Director is hereby authorized to ~~charge any fees which may be required, or~~ credit any overpayment, to Deposit Account No. **08-1935**

☒ If an additional extension of time is required, please consider this a petition therefor and charge any additional fees which may be required to Deposit Account No. **08-1935**

Blanche E. Schiller
Signature

Dated: **July 10, 2003**

Blanche E. Schiller, Esq.
Reg. No. 35,670
HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

I certify that this document and fee is being deposited on July 10, 2003 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Blanche E. Schiller
Signature of Person Mailing Correspondence

Blanche E. Schiller
Typed or Printed Name of Person Mailing Correspondence

07/16/2003 FFAHNEIA 00000099 09337158
01 FC:1251 110.00 OP

cc:

P12LARGE/REV06

AMENDMENT TRANSMITTAL LETTER (Large Entity)	Docket No.
Applicant(s): Elko et al.	PO9-99-007

Serial No.	Filing Date	Examiner	Group Art Unit
09/337,158	06/21/99	Ilwoo Park	2182

Invention: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

OIPE JC95 JUL 1 4 2003 PATENT & TRADEMARK OFFICE

OIPE JC95 JUL 1 4 2003 PATENT & TRADEMARK OFFICE

RECEIVED JUL 2 2 2003 Technology Center 2100

TO THE COMMISSIONER FOR PATENTS:

Transmitted herewith is an amendment in the above-identified application.
The fee has been calculated and is transmitted as shown below.

CLAIMS AS AMENDED

	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST # PREV. PAID FOR		NUMBER EXTRA CLAIMS PRESENT		RATE	ADDITIONAL FEE
TOTAL CLAIMS	38	-	38	=	0	x	$18.00	$0.00
INDEP. CLAIMS	5	-	5	=	0	x	$84.00	$0.00
Multiple Dependent Claims (check if applicable) ☐								$0.00
TOTAL ADDITIONAL FEE FOR THIS AMENDMENT								$0.00

☒ No additional fee is required for amendment.
☐ Please charge Deposit Account No. in the amount of
☐ A check in the amount of to cover the filing fee is enclosed.
☒ The Director is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 09-0463 (IBM)
- ☒ Any additional filing fees required under 37 C.F.R. 1.16.
- ☒ Any patent application processing fees under 37 CFR 1.17.

Blanche E. Schiller
Signature

Dated: July 10, 2003

Blanche E. Schiller, Esq.
Reg. No. 35,670
Heslin Rothenberg Farley & Mesiti P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

I certify that this document and fee is being deposited on July 10, 2003 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Blanche E. Schiller
Signature of Person Mailing Correspondence

Blanche E. Schiller
Typed or Printed Name of Person Mailing Correspondence

cc:

P11LARGE/REV06

#13
Bay
7-23-03

JUL 1 4 2003
O I P E
JC95
PATENT & TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RECEIVED
JUL 2 2 2003
Technology Center 2100

Applicants:	Elko et al.	Confirmation No.: 8602	
Serial No.:	09/337,158	Group Art Unit: 2182	
Filed:	06/21/99	Examiner: Ilwoo Park	

Title: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on July 10, 2003.

Blanche E. Schiller
Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

7/23/03

Date of Signature: July 10, 2003.

To: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Response To Office Action

Dear Sir:

This Response to Office Action is being submitted in reply to the Office Action mailed March 19, 2003, for the above-referenced patent application. A response to the Office Action was initially due on or before June 19, 2003, and thus, a Request For Extension Of Time and the requisite fee are enclosed herewith. Therefore, this Response is being timely filed.

Remarks

Reconsideration of the application and allowance of all pending claims are respectfully requested. Claims 1-13, 16, 18, 19, 23- 34, 37 and 39- 47 are pending.

In the Office Action dated March 19, 2003, claims 1, 3-13, 16, 19, 23, 25-34, 37, 39 and 41-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Wanish, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pages 3819-3820, January, 1981. Further, claims 2, 18, 24 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wanish in view of Sugiyama (U.S. Patent No. 5,933,625). Applicants respectfully, but most strenuously, traverse these rejections for the reasons herein.

In one aspect, applicants' invention is directed to generating a sequence value that is unique across multiple operating system images and is usable as a current time-of-day value. That is, the time-of-day value reflects the time at which time is requested. The sequence value is, for instance, monotonically increasing in value in that two different values resulting from two instructions either on the same CPU or different CPU's correctly imply the sequence of execution of the two instructions.

As one example, applicants claim a method of generating unique sequence values usable within a computing environment (e.g., claim 1). The method includes, for instance, providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, in which the sequence value is usable as a current time-of-day value; and including as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment. Thus, in one aspect of applicants' claimed invention, a sequence value is generated that is unique across a plurality of operating system images and is usable as a current time-of-day value. That is, the resulting sequence value still has the property of being able to be used as a time-of-day value. This is very different from the teachings of Wanish.

For example, while Wanish describes a technique for creating a unique value, this unique value is not and cannot represent a value that is usable as a current time of day value. In Wanish,

information generated from a clock is used to create an identifier; however, the identifier, once created, is not usable as a current time of day value. That is, in Wanish, the identifier is created by combining a number of different fields, one of which includes time information. However, the amalgamation of the different fields results in a value that is no longer usable as a time value. This is because in Wanish there is no requirement that the created identifier represent the current time of day and that the identifier be usable as a time of day value.

For example, there is no discussion in Wanish of ensuring that the created value is an increasing sequence value, as needed in the time of day value. Again, while a clock itself produces increasing sequence values, once that clock information is combined with other information, there is no requirement in Wanish of maintaining the increasing sequence value in the resulting value, so that it can be used for a current time-of-day value. Thus, there is no discussion in Wanish of creating a value that represents a time value, as claimed by applicants. Therefore, Wanish does not anticipate applicants' claimed invention.

In support of the rejection, the Office Action specifies paragraph 3, p. 3820 of Wanish. That section merely states that the id can be useful in establishing how long the system was operational. That is, the id includes an indication of when the id was created. From the creation date, information may be deduced. However, although timing information is provided in the value, there is no description, teaching or suggestion that the resulting id itself be usable as a current time of day value. For example, assume the technique of Wanish is used to create 2 ids for 2 different operating systems. Assume the first id is created on July 8 at 10:00 in the morning and the other is created on July 8 at 10:30 in the morning. To be a time of day value, the second value is to have a greater value than the first. However, there is no such requirement in Wanish, and since information other than time information is also used to create the ids, it is very possible that the ids do not follow in sequence order. Take the following example: using the various information to create the ids, one possible value for the first id is: 123456707081000, and a possible value for the second id is 012345607081030. Thus, while the second id was created after the first, it has a smaller value than the first. Thus, the ids cannot be relied on for time of day values. There is no requirement in Wanish that the resulting id be usable as a time of day representation. Wanish does not even use the whole clock in its this value. Thus, applicants

respectfully submit that the ids of Wanish are not usable as time of day values, as claimed by applicants, and thus, Wanish does not anticipate applicants' claimed invention.

Moreover, applicants respectfully submit that Wanish is a precursor to the Cwiakala patent that was previously cited and overcome by applicants. For example, Cwiakala describes a token format having a CPU address, CPU serial number, CPU model number, time-of-day clock (time of creation of the id). Similarly, the Wanish identifier includes a CPU address, a CPU id number, CPU model number, first half of a time-of-day clock (time of creation of the token). The two differences between Wanish and Cwiakala is that Cwiakala has a CPU serial number versus a CPU Id number of Wanish. Applicants respectfully submit that the serial number in the ESA/390 architecture is similar to the CPU Id number in the s370 architecture of IBM and that they perform the same function. Another difference is that in Cwiakala 8 bytes of the time-of-day clock was used instead of the first 4 bytes of the time-of-day clock in Wanish. Thus, the token of Cwiakala is an updated version of the host identifier of Wanish. Thus, Wanish does not anticipate applicants' claimed invention for the same reasons that Cwiakala does not anticipate applicants' claimed invention. Again, these reasons include that the identifier provided by Wanish (and Cwiakala) cannot be used as a time-of-day value, as described above. While information is taken from a timestamp in order to make a unique value, the value once created is no longer in and of itself, a timestamp or time value. That is, the value generated in Wanish, which has time information embedded therein, is not a value that is usable as a current time-of-day value.

Based on the foregoing, applicants respectfully submit that claim 1, as well as the other independent claims, are patentable over Wanish. Further, the dependent claims are patentable for the same reasons as the independent claims, as well as for their own additional features. Applicants respectfully submit that Sugiyama does not overcome the deficiencies of Wanish. Sugiyama makes no mention whatsoever of uniqueness across different operating system images.

Based on the foregoing, applicants respectfully request an indication of allowance of all pending claims. Applicants urge the Examiner to contact applicants' representative should the Examiner still have further concerns regarding this application.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: July 10, 2003.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO
09/337,158	06/21/1999	DAVID ARLEN ELKO	PO9-99-007	8602

7590 09/29/2003

BLANCHE E SCHILLER ESQ
HESLIN & ROTHENBERG P C
5 COLUMBIA CIRCLE
ALBANY, NY 12203

EXAMINER
PARK, ILWOO

ART UNIT	PAPER NUMBER
2182	14

DATE MAILED: 09/29/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev 07-01)

Office Action Summary	Application No.	Applicant(s)
	09/337,158	ELKO ET AL.
	Examiner	Art Unit
	Ilwoo Park	2182

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 July 2003* .

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-13,16,18,19,23-34,37 and 39-47* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-13,16,18,19,23-34,37 and 39-47* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1 85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. Applicants' response to office action has been filed on 7/14/2003 and reviewed.

2. Claims 1-13, 16, 18-19, 23-34, 37, and 39-47 are presented for examination.

3. Wanish and Sugiyama were cited as prior art in the last office action.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1, 3-13, 16, 19, 23, 25-34, 37, 39, and 41-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Wanish, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pages 3819-3820, January, 1981.

As to claims 1, 16, 23, 37, and 39, Wanish teaches a method of generating unique sequence values [unique host identifiers] usable within a computing environment, said method comprising:

providing as one part [time of day clock] of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable [3rd paragraph in page 3820] as a current time of day value; and

including as another part [CPU address] of said sequence value selected information usable in making said sequence value unique across [2nd paragraph in page 3819] a plurality of operating system images on one or more processors of said computing environment [MV environments].

6. As to claims 3, 19, 25, and 41, Wanish teaches providing as a further part of said sequence value a processor identifier [CPU identifier number].

7. As to claims 4, 8, and 26, Wanish teaches said providing and said including are performed by an instruction [STAP and STCK instructions].

8. As to claims 5, 10, 13, 27, 31, 34, 42, 43, and 45, Wanish teaches retrieving, by said instruction, said timing information from a physical clock and retrieving said selected information from a storage [3rd paragraph and the table for the storage format in page 3819].

9. As to claims 6 and 28, Wanish teaches said storage area is a programmable register by a set register instruction [3rd paragraph and the table for the storage format in page 3819].

10. As to claims 7, 29, and 44, Wanish teaches initializing said physical clock independently of setting said programmable register [page 3819].

11. As to claims 9 and 30, Wanish teaches initializing said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value [3rd paragraph in page 3819].

12. As to claims 11, 32, and 46, Wanish teaches initializing said physical clock to a predetermined value using a set instruction [3rd paragraph in page 3819].

13. As to claims 12, 33, and 47, Wanish teaches obtaining said selected information from a programmable register independent from said physical clock and said set instruction [page 3819].

Claim Rejections - 35 USC § 103

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. Claims 2, 18, 24, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wanish as applied to claims 1, 16, 23, and 39 above, and further in view of Sugiyama, US patent No. 5,933,625.

As to claims 2, 18, 24, and 40, Sugiyama teaches a method of generating unique sequence values usable within a computing environment, said method comprising providing as a further part of sequence value a placeholder value [col. 9, line 63-col. 10, line 3] usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide timing information of said sequence value wraps back to zero.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Wanish and Sugiyama because they both teach a method of generating unique sequence values using a timing information from a physical clock and the Sugiyama's teaching of including as a part of the sequence value the placeholder value would further increase uniqueness of the Wanish's sequence values.

Response to Arguments

16. Applicant's arguments filed 7/14/2003 have been fully considered but they are not persuasive. In the remarks, applicants argued in substance that a) Wanish's sequence value cannot be usable as a current time of day value.

For the point a), as Wanish discloses, "The last are where the ID can be useful is in analyzing operating systems dumps. Since the ID contains the time and date, it could establish how long the system was operation.", in page 3820, the ID can be usable as a current time of day value at the current time the value created; and whenever created, the ID indicates the current time of day value at the time of the creation.

Conclusion

17. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ilwoo Park whose telephone number is (703) 308-7811.

The examiner can normally be reached on Monday through Friday from 9:00 AM to 5:30 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey A Gaffin can be reached on (703) 308-3301. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA, 4th Floor (Receptionist).

JEFFREY GAFFIN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Ilwoo Park

September 25, 2003

Notice of References Cited	Application/Control No. 09/337,158	Applicant(s)/Patent Under Reexamination ELKO ET AL.	
	Examiner Ilwoo Park	Art Unit 2182	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-5933625	8/1999	Sugiyama	713/503
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
*	U	P. J. Wanish, January 1981, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pp 3819-3820
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign

#18
3/16-d
OW

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) (Large Entity)	Docket No. PO9-99-007

In Re Application Of: **Elko et al.**

OIPE
MAR 0 3 2004
PATENT & TRADEMARK OFFICE

Serial No.	Filing Date	Examiner	Group Art Unit
09/337,158	06/21/99	Ilwoo Park	2182

Invention: **METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES**

RECEIVED
MAR 0 5 2004
Technology Center 2100

TO THE COMMISSIONER FOR PATENTS:

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response to the Office Action of **September 29, 2003** (Date) above-identified application.

The requested extension is as follows (check time period desired):

☐ One month ☒ Two months ☐ Three months ☐ Four months ☐ Five months

from: **December 29, 2003** (Date) until: **February 29, 2004 (Sunday)** (Date)

The fee for the extension of time is **$420** and is to be paid as follows:

☒ A check in the amount of the fee is enclosed.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account No.

☒ If an additional extension of time is required, please consider this a petition therefor and charge any additional fees which may be required to Deposit Account No. **08-1935**

Blanche E. Schiller
Signature

Dated: **March 01, 2004**

Blanche E. Schiller, Esq.
Reg. No. 35,670
HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

2004 MHORDSFI 00000005 090463 09337158
420.00 OP

I certify that this document and fee is being deposited on **March 01, 2004** with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Commissioner for Patents, P O. Box 1450, Alexandria, VA 22313-1450.

Jill K. Becker
Signature of Person Mailing Correspondence

Jill K. Becker
Typed or Printed Name of Person Mailing Correspondence

cc:

P12LARGE/REV06

RCE/2700

PTO/SB/30 (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

OIPE MAR 03 2004 PATENT & TRADEMARK OFFICE

3-16-04

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Address to
Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	09/337,158
Filing Date	06/21/99
First Named Inventor	David A. Elko
Art Unit	2182
Examiner Name	Ilwoo Park
Attorney Docket Number	PO9-99-007

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. Submission required under 37 CFR 1.114 — Note: If the RCE is proper, any previously filed unentered and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).
   a. ☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.
      i. ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on ________
      ii. ☐ Other ________

RECEIVED
MAR 05 2004
Technology Center 2100

   b. ☒ Enclosed
      i. ☒ Amendment/Reply
      ii. ☐ Affidavit(s)/Declaration(s)
      iii. ☐ Information Disclosure Statement (IDS)
      iv. ☐ Other ________

2. Miscellaneous
   a. ☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of ______ months. (Period of suspension shall not exceed 3 months; Fee under 37 CFR 1 17(i) required)
   b. ☐ Other ________

3. Fees — The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.
   a. ☒ The Director is hereby authorized to charge the following fees, or credit any overpayments, to Deposit Account No. 09-0463 (IBM)
      i. ☒ RCE fee required under 37 CFR 1.17(e)
      ii. ☐ Extension of time fee (37 CFR 1.136 and 1.17) 03/05/2004 AWONDAF1 00000065 090463 09337158
      iii. ☐ Other ________ ~~02 FC:1801 770.00 DA~~
   b. ☐ Check in the amount of $ ________ enclosed
   c. ☐ Payment by credit card (Form PTO-2038 enclosed)

   WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Name (Print / Type)	Blanche E. Schiller, Esq.	Registration No. (Attorney / Agent)	35,670
Signature	Blanche E. Schiller	March 01, 2004	

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop RCE, Commissioner For Patents, P.O. Box 1450, Alexandria, VA 22313-1450 or facsimile transmitted to the U.S. Patent and Trademark Office on the date shown below.

Name (Print / Type)	Jill K. Becker		
Signature	Jill K. Becker	Date	March 01, 2004

This collection of information is required by 37 CFR 1 114 The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1 14 This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing the burden, should be sent to the Chief Information Officer, U S. Patent and Trademark Office, U S Department of Commerce, P O Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO: **Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

#20D
3-16-04

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Elko et al.	Confirmation No.:	8602
Serial No.:	09/337,158	Group Art Unit:	2182
Filed:	06/21/99	Examiner:	Ilwoo Park
Title:	METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES		

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on March 01, 2004.

Jill K. Becker
Jill K. Becker

Date of Signature: March 01, 2004

RECEIVED
MAR 0 5 2004
Technology Center 2100

To: Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Preliminary Amendment

Dear Sir:

Prior to examination of the above-referenced application, please amend the application, as follows:

Amendment to the Claims

Kindly cancel claim 50, add new claims 51-53, and amend claims 1, 16, 23, 37, 39, as set forth below. In compliance with the Revised Amendment Format published in the Official Gazette on February 25, 2003, a complete listing of claims is provided herein. The changes in the amended claims are shown by strikethrough (for deleted matter) and underlining (for added matter), with the exception that double brackets are used to indicate deleted matter if strikethrough cannot be easily perceived.

1. (Currently Amended) A method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information~~, said sequence value being usable as a current time of day value~~; and

including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment<u>, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment</u>.

2. (Original) The method of claim 1, further comprising providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

3. (Original) The method of claim 1, further comprising providing as a further part of said sequence value a processor identifier.

4. (Original) The method of claim 1, wherein said providing and said including are performed by an instruction.

5. (Original) The method of claim 4, wherein said providing comprises retrieving, by said instruction, said timing information from a physical clock, and wherein said including comprises retrieving, by said instruction, said selected information from a storage area.

6. (Original) The method of claim 5, wherein said storage area is a programmable register set by a set register instruction.

7. (Original) The method of claim 6, further comprising initializing said physical clock independently of setting said programmable register.

8. (Original) The method of claim 4, wherein said instruction is a STORE CLOCK EXTENDED instruction.

9. (Original) The method of claim 4, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

10. (Original) The method of claim 1, further comprising receiving said timing information from a physical clock of said computing environment.

11. (Original) The method of claim 10, further comprising initializing said physical clock to a predefined value using a set instruction.

12. (Original) The method of claim 11, further comprising obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

13. (Original) The method of claim 1, further comprising obtaining said selected information from a programmable register set by a set register instruction.

14. (Canceled)

15. (Canceled)

17 ~~16~~. (Currently Amended) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and providing a value usable as a current time of day clock value in real-time processing by one or more processors of the computing environment, said representation comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to make the value unique across a plurality of operating system images on one or more processors of said computing environment.

17. (Canceled)

18. (Previously Presented) The memory of claim ~~16~~ 17, wherein said representation further comprises a placeholder component usable in ensuring that said value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

19. (Original) The memory of claim ~~16~~ 17, wherein said representation further comprises a processor identifier component including processor information.

20. (Canceled)

21. (Canceled)

22. (Canceled)

20 ~~23~~. (Currently Amended) A system of generating unique sequence values usable within a computing environment, said system comprising:

means for providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information~~, said sequence value being usable as a current time of day value~~; and

means for including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment.

21 ~~24~~. (Original) The system of claim ~~23~~ 20, further comprising means for providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

22 ~~25~~. (Original) The system of claim ~~23~~ 20, further comprising means for providing as a further part of said sequence value a processor identifier.

23 ~~26~~. (Original) The system of claim ~~23~~ 20, wherein said means for providing and said means for including comprise an instruction.

24 ~~27~~. (Original) The system of claim ~~26~~ 23, wherein said means for providing comprises means for retrieving, by said instruction, said timing information from a physical clock, and wherein said means for including comprises means for retrieving, by said instruction, said selected information from a storage area.

25 ~~28~~. (Original) The system of claim 24 ~~27~~, wherein said storage area is a programmable register set by a set register instruction.

26 ~~29~~. (Original) The system of claim 25 ~~28~~, further comprising means for initializing said physical clock independently of setting said programmable register.

27 ~~30~~. (Original) The system of claim 23 ~~26~~, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

28 ~~31~~. (Original) The system of claim 20 ~~23~~, further comprising means for receiving said timing information from a physical clock of said computing environment.

29 ~~32~~. (Original) The system of claim 28 ~~31~~, further comprising means for initializing said physical clock to a predefined value using a set instruction.

30 ~~33~~. (Original) The system of claim 29 ~~32~~, further comprising means for obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

31 ~~34~~. (Original) The system of claim 20 ~~23~~, further comprising means for obtaining said selected information from a programmable register set by a set register instruction.

35. (Canceled)

36. (Canceled)

32 ~~37~~. (Currently Amended) A system of generating unique sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information~~, said sequence value being usable as a current time of day value~~; and

said processor being further adapted to provide as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment.

38. (Canceled)

33 ~~39~~. (Currently Amended) An article of manufacture, comprising:

D1

at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising:

computer readable program code means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information~~, said sequence value being usable as a current time of day value~~; and

computer readable program code means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment.

D

34 ~~40~~. (Original) The article of manufacture of claim 33 ~~39~~, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

35 ~~41~~. (Original) The article of manufacture of claim 33 ~~39~~, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a processor identifier.

36 ~~42~~. (Original) The article of manufacture of claim 33 ~~39~~, wherein said computer readable program code means for causing a computer to provide comprises computer readable program code means for causing a computer to retrieve, by an instruction, said timing information from a physical clock, and wherein said computer readable program code means for causing a computer to include comprises computer readable program code means for causing a computer to retrieve, by said instruction, said selected information from a storage area.

37 ~~43~~. (Original) The article of manufacture of claim 36 ~~42~~, wherein said storage area is a programmable register set by a set register instruction.

38 ~~44~~. (Original) The article of manufacture of claim 37 ~~43~~, further comprising computer readable program code means for causing a computer to initialize said physical clock independently of setting said programmable register.

39 ~~45~~. (Original) The article of manufacture of claim 33 ~~39~~, further comprising computer readable program code means for causing a computer to receive said timing information from a physical clock of said computing environment.

40 ~~46~~. (Original) The article of manufacture of claim 39 ~~45~~, further comprising computer readable program code means for causing a computer to initialize said physical clock to a predefined value using a set instruction.

41 ~~47~~. (Original) The article of manufacture of claim 40 ~~46~~, further comprising computer readable program code means for causing a computer to obtain said selected information from a programmable register independent from said physical clock and said set instruction.

48. (Canceled)

49. (Canceled)

50. (Canceled)

14 ~~51~~. (New) The method of claim 1, wherein the sequence value that is generated is considered as one entity, said one entity being usable as the current time of day clock value.

15 ~~52~~. (New) The method of claim 1, wherein said sequence value indicates a correct sequence of execution of an instruction, regardless of which processor of the one or more processors executes the instruction.

16 ~~53~~. (New) The method of claim 1, wherein the sequence value has a predictable resolution.

Remarks

Entry of this amendment, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. Claims 1-13, 16, 18-19, 23-34, 37, 39-47 and 51-53 are pending.

With the above amendments, applicants are clarifying that the sequence value is usable as a current time of day clock value in real-time processing. Further, applicants have added dependent claims that define additional characteristics of the sequence value that is generated. Support for these amendments can be found throughout the specification (e.g., page 2, line 15-page 3, line 17; page 11, lines 9-14; page 15, lines 20-23; page 16, lines 24-29; page 18, lines 1-7; page 21, lines 21-23; page 29, lines 8-13; page 29, lines 10-15; page 31, lines 20-29; page 36). Thus, no new matter is added.

In the Office Action dated September 29, 2003, claims 1, 3-13, 16, 19, 23, 25-34, 37, 39 and 41-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Wanish, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pages 3819-3820, January, 1981. Further, claims 2, 18, 24 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wanish in view of Sugiyama (U.S. Patent No. 5,933,625). Applicants respectfully, but most strenuously, traverse these rejections to any extent deemed applicable to the amended claims.

In one aspect, applicants' invention is directed to generating a sequence value that is unique across multiple operating system images and is used as a current time of day clock value by one or more processors of the computing environment. That is, the sequence value, as generated, reflects the time at which time is requested by a processor and the processor can use (i.e., act upon) that value in real time. The sequence value is, for instance, monotonically increasing in that two different values resulting from two instructions either on the same CPU or different CPUs correctly imply the sequence of execution of the two instructions.

As one example, applicants claim a method of generating unique sequence values usable within a computing environment (e.g., claim 1). The method includes, for instance, providing as one part of a sequence value timing information comprising at least one of time of day information and date information; and including as another part of the sequence value selected

information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment, wherein the sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment. Thus, in one aspect of applicants' claimed invention, a sequence value is generated that is unique across a plurality of operating system images and is still usable as a current time of day clock value in real-time processing. That is, the resulting sequence value, once generated, still has the property of being able to be used as a time of day value for real-time processing. This is very different from the teachings of Wanish.

Wanish describes the creation of a unique identifier, as opposed to the creation of a unique current time of day value. The identifier in Wanish includes some time information, but the identifier itself is not usable as a current time of day value in real-time processing. This is evident in Wanish by, for instance, the fact that in order to use any time information from the Wanish identifier that information needs to be extracted, since only a portion of the identifier includes the time information, unlike applicants' sequence value in which the entire value is the timing information. The extracting of the information takes time, and thus, cannot be used for real-time processing.

It is indicated in Wanish that the id can be useful in analyzing operating system dumps (Wanish, p. 3820). However, this analysis is in post-processing, not in real-time processing, as claimed by applicants in claim 1, in which it is explicitly recited that the sequence value is usable as a current time of day clock value in real-time processing. Thus, applicants respectfully submit that Wanish does not describe applicants' claimed element of generating a sequence value that is usable as a current time of day clock value in real-time processing.

Further, applicants respectfully submit that the identifier created in Wanish, i.e., the identifier, as a whole, is not used or is not usable as a current time of day value in real-time processing. While information generated from a clock is used to create the identifier in Wanish, the identifier, once created, is not usable as a current time of day value. That is, in Wanish, the identifier is created by combining a number of different fields, one of which includes time information. However, the amalgamation of the different fields results in a value that is no longer usable as a time value. This is because in Wanish there is no requirement that the created

identifier represent the current time of day and that the identifier be usable as a time of day value in real-time processing.

For example, there is no discussion in Wanish of ensuring that the created value is an increasing sequence value, as needed in the time of day value. Again, while a clock itself produces increasing sequence values, once that clock information is combined with other information, there is no requirement in Wanish of maintaining the increasing sequence value in the resulting value, so that it can be used for a current time-of-day value. For instance, assume the technique of Wanish is used to create two ids for two different operating systems. Assume the first id is created on July 8 at 10:00 in the morning and the other is created on July 8 at 10:30 in the morning. To be a time of day value, the second value is to have a greater value than the first. However, there is no such requirement in Wanish, and since information other than time information is also used to create the ids, it is very possible that the ids do not follow in sequence order. Take the following example: using the various information to create the ids, one possible value for the first id is: 123456707081000, and a possible value for the second id is 012345607081030. Thus, while the second id was created after the first, it has a smaller value than the first. Thus, the ids cannot be relied on for current time of day values. There is no requirement in Wanish that the resulting id be usable as a time of day representation. Wanish does not even use the whole clock in its value. Thus, applicants respectfully submit that the ids of Wanish are not usable as time of day values within a computing environment, as claimed by applicants, and thus, Wanish does not anticipate applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that claim 1, as well as the other independent claims, are patentable over Wanish. Further, the dependent claims are patentable for the same reasons as the independent claims, as well as for their own additional features. For example, dependent claim 51, explicitly recites that the sequence value that is generated is considered as one entity and it is that entity that is usable as the current time of day clock value. This is very different from Wanish because the resulting value in Wanish cannot be used as a current time of day clock, as described above. While Wanish includes some time information, that time information is combined with other fields and the resulting value cannot be used as a current time of day clock value. Thus, applicants respectfully submit that claim 51 is patentable over Wanish.

Dependent claims 52 and 53 also describe characteristics of applicants' sequence value that are missing from the generated identifier of Wanish. Thus, applicants respectfully submit that those claims are also patentable over Wanish.

For all of the above reasons, applicants respectfully submit that their invention is patentable over Wanish. Additionally, the other cited art does not overcome the deficiencies of Wanish. Thus, applicants respectfully request an indication of allowance of all pending claims.

Applicants urge the Examiner to contact applicants' representative should the Examiner still have further concerns regarding this application.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: March 01, 2004.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

JAN. -06' 04 (TUE) 14:04 HESLIN ROTHENBERG TEL:518 452 5579 P. 001

HESLIN ROTHENBERG FARLEY & MESITI P.C.

INTELLECTUAL PROPERTY LAW

PATENTS • TRADEMARKS • COPYRIGHTS

Robert E. Heslin
Jeff Rothenberg
Kevin P. Radigan
Susan F. Farley
Nicholas Mesiti
Philip E. Hansen*
Blanche E. Schiller
Wayne F. Reinke
David P. Miranda

*Patent Agent

Attorneys at Law
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579
www.hrfmlaw.com

Kathy Smith Dias
Mary Louise Gioeni
David A. Pascarella
Victor A. Cardona
Lee Palmateer
John Pietrangelo*
Brett M. Hutton
James M. Syta

Martha L. Boden
Jill M. Breedlove
Of Counsel

RECEIVED
CENTRAL FAX CENTER
JAN 0 6 2004
OFFICIAL

Date: January 6, 2004

From: Blanche E. Schiller, Esq. (e-mail: bes@hrfmlaw.com)

No. of Pages: 19 (Including this Page)

Fax No.: (703) 872-9306

To: Examiner Ilwoo Park
Group Art Unit 2182
U.S. Patent and Trademark Office
Alexandria, VA 22313-1450

Re: U.S. Patent Application Serial No.: 09/337,158
Applicants: Elko et al.
Attorney Docket No.: 0560.225

Message: Please find attached hereto the following documents:

1. Amendment Transmittal Letter (1 page) (in duplicate);
2. Petition for Extension of Time (1 page) (in duplicate)
3. Response to Final Office Action (14 pages).

BES:jb
Attachments

0560225-FAX01-PTO

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) (Large Entity)		Docket No. PO9-99-007
In Re Application Of: Elko et al.			
Serial No. 09/337,158	Filing Date 06/21/99	Examiner Ilwoo Park	Group Art Unit 2182

Invention: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

TO THE COMMISSIONER FOR PATENTS:

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response to the Office Action of September 29, 2003 (Date) above-identified application.

The requested extension is as follows (check time period desired):

☒ One month ☐ Two months ☐ Three months ☐ Four months ☐ Five months

from: December 29, 2003 (Date) until: January 29, 2004 (Date)

The fee for the extension of time is $110 and is to be paid as follows:

☐ A check in the amount of the fee is enclosed.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account No. 08-1935

☒ If an additional extension of time is required, please consider this a petition therefor and charge any additional fees which may be required to Deposit Account No. 08-1935

Blanche E. Schiller (Signature) Dated: January 6, 2004

Blanche E. Schiller, Esq.
Reg. No. 35,670
HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

I hereby certify that this correspondence is being transmitted by facsimile transmission to:

Examiner Ilwoo Park, Group Art Unit 2182, U. S. Patent and Trademark Office, Alexandria, VA 22313-1450, Facsimile No. 703-872-9306, on January 6, 2004.

Jill K. Becker

Jill K. Becker

cc:

AMENDMENT TRANSMITTAL LETTER (Large Entity) Applicant(s): Elko et al.		Docket No. PO9-99-007

Serial No.	Filing Date	Examiner	Group Art Unit
09/337,158	06/21/99	Ilwoo Park	2182

Invention: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

TO THE COMMISSIONER FOR PATENTS:

Transmitted herewith is an amendment in the above-identified application.

The fee has been calculated and is transmitted as shown below.

CLAIMS AS AMENDED

	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST # PREV. PAID FOR		NUMBER EXTRA CLAIMS PRESENT		RATE	ADDITIONAL FEE
TOTAL CLAIMS	39	-	38	=	1	x	$18.00	$18.00
INDEP. CLAIMS	5	-	5	-	0	x	$84.00	$0.00
Multiple Dependent Claims (check if applicable) ☐								$0.00
TOTAL ADDITIONAL FEE FOR THIS AMENDMENT								$18.00

☐ No additional fee is required for amendment.

☒ Please charge Deposit Account No. 09-0463 (IBM) in the amount of $18.00

☐ A check in the amount of to cover the filing fee is enclosed.

☒ The Director is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No 09-0463 (IBM)

- ☒ Any additional filing fees required under 37 C.F.R. 1.16.
- ☒ Any patent application processing fees under 37 CFR 1.17.

Blanche E. Schiller
Signature

Dated: January 6, 2004

Blanche E. Schiller, Esq.
Reg. No. 35,670
Heslin Rothenberg Farley & Mesiti P.C.
5 Columbia Circle
Albany, NY 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

I hereby certify that this correspondence is being transmitted by facsimile transmission to:

Examiner Ilwoo Park, Group Art Unit 2182, U. S. Patent and Trademark Office, Alexandria, VA 22313-1450, Facsimile No. 703-872-9306, on January 6, 2004.

Jill K. Becker

Jill K. Becker

cc:

OFFICIAL

RESPONSE UNDER 37 C.F.R. 1.116
EXPEDITED PROCEDURE
EXAMINING GROUP 2182

RECEIVED
CENTRAL FAX CENTER
JAN 06 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

AF 216 Amdt C (ne) 91 1/15/04

Applicants:	Elko et al.	Confirmation No.:	8602
Serial No.:	09/337,158	Group Art Unit:	2182
Filed:	06/21/99	Examiner:	Ilwoo Park

Title: METHOD, SYSTEM AND PROGRAM PRODUCTS FOR GENERATING SEQUENCE VALUES THAT ARE UNIQUE ACROSS OPERATING SYSTEM IMAGES

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that this correspondence is being transmitted by facsimile transmission to: Examiner Ilwoo Park, Group Art Unit 2182, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450, Facsimile No. (703) 872-9306, on January 6, 2004.

Jill K. Becker

Jill K. Becker

Date of Signature: January 6, 2004.

To: Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Response To Final Office Action

Dear Sir:

This Response to Office Action is being submitted in reply to the Office Action mailed September 29, 2003 for the above-referenced patent application. A response to the Office Action was initially due on or before December 29, 2003, and thus, a Request For Extension Of Time and the requisite fee are enclosed herewith. Therefore, this Response is being timely filed.

Amendment to the Claims

Kindly add claim 50, as set forth below. In compliance with the Revised Amendment Format published in the Official Gazette on February 25, 2003, a complete listing of claims is provided herein. The changes in the amended claims are shown by strikethrough (for deleted matter) and underlining (for added matter).

1. (Previously Presented) A method of generating unique sequence values usable within a computing environment, said method comprising:

providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

2. (Original) The method of claim 1, further comprising providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

3. (Original) The method of claim 1, further comprising providing as a further part of said sequence value a processor identifier.

4. (Original) The method of claim 1, wherein said providing and said including are performed by an instruction.

5. (Original) The method of claim 4, wherein said providing comprises retrieving, by said instruction, said timing information from a physical clock, and wherein said including comprises retrieving, by said instruction, said selected information from a storage area.

6. (Original) The method of claim 5, wherein said storage area is a programmable register set by a set register instruction.

7. (Original) The method of claim 6, further comprising initializing said physical clock independently of setting said programmable register.

8. (Original) The method of claim 4, wherein said instruction is a STORE CLOCK EXTENDED instruction.

Sub By

9. (Original) The method of claim 4, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

10. (Original) The method of claim 1, further comprising receiving said timing information from a physical clock of said computing environment.

11. (Original) The method of claim 10, further comprising initializing said physical clock to a predefined value using a set instruction.

12. (Original) The method of claim 11, further comprising obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

13. (Original) The method of claim 1, further comprising obtaining said selected information from a programmable register set by a set register instruction.

14. Canceled

15. Canceled

16. (Previously Presented) A memory for storing data, said memory comprising:

a representation of a time-of-day clock, said representation being usable within a computing environment and providing a value usable as a current time of day value, said representation comprising:

a timing component comprising timing information including at least one of time-of-day information and date information; and

a programmable field component comprising selected information to make the value unique across a plurality of operating system images on one or more processors of said computing environment.

17. Canceled

18. (Previously Presented) The memory of claim 16, wherein said representation further comprises a placeholder component usable in ensuring that said value is an increasing sequence value, even when a physical clock used to provide said timing information wraps back to zero.

19. (Original) The memory of claim 16, wherein said representation further comprises a processor identifier component including processor information.

20. Canceled

21. Canceled

22. Canceled

23. (Previously Presented) A system of generating unique sequence values usable within a computing environment, said system comprising:

means for providing as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

means for including as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

24. (Original) The system of claim 23, further comprising means for providing as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

25. (Original) The system of claim 23, further comprising means for providing as a further part of said sequence value a processor identifier.

26. (Original) The system of claim 23, wherein said means for providing and said means for including comprise an instruction.

27. (Original) The system of claim 26, wherein said means for providing comprises means for retrieving, by said instruction, said timing information from a physical clock, and wherein said means for including comprises means for retrieving, by said instruction, said selected information from a storage area.

28. (Original) The system of claim 27, wherein said storage area is a programmable register set by a set register instruction.

29. (Original) The system of claim 28, further comprising means for initializing said physical clock independently of setting said programmable register.

30. (Original) The system of claim 26, wherein said instruction is issued by a program of said computing environment desiring said sequence value, and wherein said instruction is independent such that communication between said plurality of operating system images is not necessary to generate said sequence value.

31. (Original) The system of claim 23, further comprising means for receiving said timing information from a physical clock of said computing environment.

32. (Original) The system of claim 31, further comprising means for initializing said physical clock to a predefined value using a set instruction.

33. (Original) The system of claim 32, further comprising means for obtaining said selected information from a programmable register independent from said physical clock and said set instruction.

34. (Original) The system of claim 23, further comprising means for obtaining said selected information from a programmable register set by a set register instruction.

35. Canceled

36. Canceled

37. (Previously Presented) A system of generating unique sequence values usable within a computing environment, said system comprising:

a processor adapted to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

said processor being further adapted to provide as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

38. Canceled

39. (Previously Presented) An article of manufacture, comprising:

at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising:

computer readable program code means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information, said sequence value being usable as a current time of day value; and

computer readable program code means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment.

40. (Original) The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a placeholder value usable in ensuring that said sequence value is an increasing sequence value, even when a physical clock used to provide said timing information of said sequence value wraps back to zero.

41. (Original) The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to provide as a further part of said sequence value a processor identifier.

42. (Original) The article of manufacture of claim 39, wherein said computer readable program code means for causing a computer to provide comprises computer readable program code means for causing a computer to retrieve, by an instruction, said timing information from a physical clock, and wherein said computer readable program code means for causing a computer to include comprises computer readable program code means for causing a computer to retrieve, by said instruction, said selected information from a storage area.

43. (Original) The article of manufacture of claim 42, wherein said storage area is a programmable register set by a set register instruction.

44. (Original) The article of manufacture of claim 43, further comprising computer readable program code means for causing a computer to initialize said physical clock independently of setting said programmable register.

5350.

45. (Original) The article of manufacture of claim 39, further comprising computer readable program code means for causing a computer to receive said timing information from a physical clock of said computing environment.

46. (Original) The article of manufacture of claim 45, further comprising computer readable program code means for causing a computer to initialize said physical clock to a predefined value using a set instruction.

47. (Original) The article of manufacture of claim 46, further comprising computer readable program code means for causing a computer to obtain said selected information from a programmable register independent from said physical clock and said set instruction.

48. Canceled

55

49. Canceled

50. (New) The method of claim 1, wherein the sequence value is used as the current time of day value in real-time processing by one or more processors of the computing environment.

Remarks

Entry of the amendment, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. Claims 1-13, 16, 18, 19, 23- 34, 37, 39- 47 and 50 are pending.

In a bona fide attempt to further prosecution of this application, applicants have added a dependent claim, which applicants respecrfully request be entered and considered by the Examiner. In particular, claim 50 has been added to further clarify what is meant by "usable as a current time of day value." As used herein and throughout applicants' specification, and as known in the art, "usable as a current time of day value" means that the current time of day value can be used by processors in real-time processing. For example, the generated value is the system clock (i.e., a current time of day value) used in real-time processing that needs a system clock. Applicants did not provide this particular amendment earlier, since they believed that the phrase "usable as a current time of day value" was clear in this manner. However, applicants are offering this amendment at this time as a clarification, and again, as a bona fide attempt to further prosecution of this application.

Support for this amendment can be found throughout the specification (e.g., p. 11, lines 9-14; p. 15, lines 20-23, p. 16, lines 24-29; p. 18, lines 1-7; p. 21, lines 21-23; p. 29, lines 8-13; p. 29, lines 10-15; p. 31, lines 20-29; p. 36). Thus, no new matter is added.

In the Office Action dated September 29, claims 1, 3-13, 16, 19, 23, 25-34, 37, 39 and 41-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Wanish, IBM Technical Disclosure Bulletin, Vol. 23, No. 8, pages 3819-3820, January, 1981. Further, claims 2, 18, 24 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wanish in view of Sugiyama (U.S. Patent No. 5,933,625). Applicants respectfully, but most strenuously, traverse these rejections for the reasons herein.

In one aspect, applicants' invention is directed to generating a sequence value that is unique across multiple operating system images and is used as a current time of day value by one or more processors of the computing environment. That is, the sequence value, as generated, reflects the time at which time is requested by a processor and the processor can use (i.e., act

upon) that value in real-time. The sequence value is, for instance, monotonically increasing in that two different values resulting from two instructions either on the same CPU or different CPU's correctly imply the sequence of execution of the two instructions.

As one example, applicants claim a method of generating unique sequence values usable within a computing environment (e.g., claim 1). The method includes, for instance, providing as one part of a sequence value timing information comprising at least one of time of day information and date information, in which the sequence value is usable as a current time of day value; and including as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment. Thus, in one aspect of applicants' claimed invention, a sequence value is generated that is unique across a plurality of operating system images and is usable as a current time of day value. That is, the resulting sequence value still has the property of being able to be used as a time of day value for real-time processing. This is very different from the teachings of Wanish.

Wanish describes the creation of a unique identifier, as opposed to the creation of a unique current time of day value. The identifier in Wanish includes some time information, but the identifier itself is not usable as a current time of day value. That is, the identifier in Wanish is not used as the current time of day value in real-time processing. This is evident in Wanish by, for instance, the fact that in order to use any time information from the Wanish identifier that information needs to be extracted, since only a portion of the identifier includes the time information, unlike applicants' sequence value in which the entire value is the timing information. The extracting of the information takes time, and thus, cannot be used for real-time processing.

It is indicated in Wanish that the id can be useful in analyzing operating system dumps (Wanish, p. 3820). However, this analysis is in post-processing, not in real-time processing, as claimed by applicants in claim 1, in which the words "current time of day value" are used and as explicitly claimed in dependent claim 50. Thus, applicants respectfully submit that Wanish does not describe applicants' claimed element of generating a sequence value that is usable as a current time of day value.

Further, applicants respectfully submit that the identifier created in Wanish, i.e., the identifier, as a whole, is not used or is not usable as a time of day value. While information generated from a clock is used to create the identifier in Wanish, the identifier, once created, is not usable as a current time of day value. That is, in Wanish, the identifier is created by combining a number of different fields, one of which includes time information. However, the amalgamation of the different fields results in a value that is no longer usable as a time value. This is because in Wanish there is no requirement that the created identifier represent the current time of day and that the identifier be usable as a time of day value.

For example, there is no discussion in Wanish of ensuring that the created value is an increasing sequence value, as needed in the time of day value. Again, while a clock itself produces increasing sequence values, once that clock information is combined with other information, there is no requirement in Wanish of maintaining the increasing sequence value in the resulting value, so that it can be used for a current time-of-day value. For instance, assume the technique of Wanish is used to create two ids for two different operating systems. Assume the first id is created on July 8 at 10:00 in the morning and the other is created on July 8 at 10:30 in the morning. To be a time of day value, the second value is to have a greater value than the first. However, there is no such requirement in Wanish, and since information other than time information is also used to create the ids, it is very possible that the ids do not follow in sequence order. Take the following example: using the various information to create the ids, one possible value for the first id is: 1234567070081000, and a possible value for the second id is 0123456070081030. Thus, while the second id was created after the first, it has a smaller value than the first. Thus, the ids cannot be relied on for current time of day values. There is no requirement in Wanish that the resulting id be usable as a time of day representation. Wanish does not even use the whole clock in its value. Thus, applicants respectfully submit that the ids of Wanish are not usable as time of day values within a computing environment, as claimed by applicants, and thus, Wanish does not anticipate applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that claim 1, as well as the other independent claims, are patentable over Wanish. Further, the dependent claims are patentable for the same reasons as the independent claims, as well as for their own additional features.

For example, in dependent claim 50, applicants explicitly recite that the generated unique sequence value is used as the current time of day value in real-time processing. There is no description, teaching or suggestion in Wanish of the generated value, as a whole, being a time value. Further, there is no description, teaching or suggestion of the created identifier being used as the current time of day value in real-time processing. Again, while the identifier includes time information, the combined fields of the generated identifier cannot be used as a time value in real-time processing. This is because there is no teaching or suggestion of creating a value that has current time of day properties, such as monotonicity and predictable resolution. Again, take the following example in which the technique of Wanish is used to create two ids for two different operating systems: Assume the first id is created on July 8 at 10:00 in the morning and the other is created on July 8 at 10:30 in the morning. To be a time of day value, the second value is to have a greater value than the first. However, there is no such requirement in Wanish, and since information other than time information is also used to create the ids, it is very possible that the ids do not follow in sequence order. Take the following example: using the various information to create the ids, one possible value for the first id is: 123456707081000, and a possible value for the second id is 0123456070810030. Thus, while the second id was created after the first, it has a smaller value than the first. Thus, the ids cannot be relied on for time of day values or system clocks.

Since the value of Wanish is not and cannot be used as a current time of day value used in real-time processing, applicants respectfully submit that Wanish does not anticipate applicants' claimed invention, and thus, respectfully request an indication of allowability of dependent claim 50.

Based on the foregoing, applicants respectfully submit that their invention is patentable over Wanish. Additionally, the other cited art does not overcome the deficiencies of Wanish. Thus, applicants respectfully request an indication of allowance of all pending claims.

Applicants urge the Examiner to contact applicants' representative should the Examiner still have further concerns regarding this application.

Respectfully submitted,

Blanche E. Schiller
Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: January 6, 2004.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

Merriam-Webster
Merriam-Webster OnLine

Also Visit: | Unabridged | Encyclopedia Britannica | Visual NEW ESL: | Learner's for Kids: | Spell It! NEW

Dictionary Thesaurus Spanish/English Medical

Home
Visit Our Sites
Premium Services
Downloads
Word of the Day
Word Games
Open Dictionary
Spelling Bee Hive
Word for the Wise
Online Store
Help
About Us

va=routine

2 entries found.

routine[1,noun]
routine[2,adjective]

Main Entry: [1]**rou·tine**
Pronunciation: \rü-ˈtēn\
Function: *noun*
Etymology: French, from Middle French, from *route* traveled way
Date: 1676

1 a : a regular course of procedure <if resort to legal action becomes a campus *routine* — J. A. Perkins> **b :** habitual or mechanical performance of an established procedure <the *routine* of factory work>
2 : a reiterated speech or formula <the old "After you" *routine* — Ray Russell>
3 : a worked-out part (as of an entertainment or sports contest) that may be often repeated <a dance *routine*> <a gymnastic *routine*>; *especially* **:** a theatrical number
4 : a sequence of computer instructions for performing a particular task

Learn more about "routine" and related topics at Britannica.com

Find Jobs in Your City

Pronunciation Symbols

Share this entry:

Link to this page:
routine

Cite this page:

MLA Style
"routine." Merriam-Webster Online Dictionary. 2008.
Merriam-Webster Online. 30 May 2008
<http://www.merriam-webster.com/dictionary/routine>

APA Style
routine. (2008). In *Merriam-Webster Online Dictionary*.
Retrieved May 30, 2008, from http://www.merriam-webster.com/dictionary/routine

Browse the Dict
A B C D E F G H
Z #

Products Premium Services Company Info Contact Us Advertising Info Privacy Policy

IEEE 100

THE AUTHORITATIVE DICTIONARY OF IEEE STANDARDS TERMS

SEVENTH EDITION

IEEE
Published by
Standards Information Network
IEEE Press

Trademarks and disclaimers

IEEE believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. IEEE is not responsible for any inadvertent errors.

Other tradenames and trademarks in this document are those of their respective owners.

The Institute of Electrical and Electronics Engineering, Inc.
3 Park Avenue, New York, NY, 10016-5997, USA

To order IEEE Press publications, call 1-800-678-IEEE.

Print: ISBN 0-7381-2601-2 *SP1122*

See other standards and standards-related product listings at: http://standards.ieee.org/

The publisher believes that the information and guidance given in this work serve as an enhancement to users, all parties must rely upon their own skill and judgement when making use of it. The publisher does not assume any liability to anyone for any loss or damage caused by any error or omission in the work, whether such error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

This work is published with the understanding that the IEEE is supplying information through this publication, not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought. The IEEE is not responsible for the statements and opinions advanced in this publication.

Library of Congress Cataloging-in-Publication Data

IEEE 100 : the authoritative dictionary of IEEE standards terms.—7th ed.
p. cm.
ISBN 0-7381-2601-2 (paperback : alk. paper)
1. Electric engineering—Dictionaries. 2. Electronics—Dictionaries. 3. Computer engineering—Dictionaries. 4. Electric engineering—Acronyms. 5. Electronics—Acronyms. 6. Computer engineering—Acronyms. I. Institute of Electrical and Electronics Engineers.

TK9 .I28 2000
621.3′03—dc21 00-050601

computer control unit *See:* instruction control unit.

computer data (software) Data available for communication between or within computer equipment. Such data can be external (in computer-readable form) or resident within the computer equipment and can be in the form of analog or digital signals. *See also:* computer. (C/SE) 729-1983s

computer database *See:* database.

computer description language *See:* hardware description language.

Computer Design Language A design language for describing or designing computer architectures at the register level. (C) 610.13-1993w

computer diagram (analog computer) A functional drawing showing interconnections between computing elements, such interconnections being specified for the solution of a particular set of equations. *See also:* computer program; problem board. (C) 165-1977w

computer equation (machine equation) (analog computer) An equation derived from a mathematical model for use on a computer which is equivalent or proportional to the original equation. *See also:* scale factor. (C) 165-1977w

computer generated force (CGF) Simulation of entities on the virtual battlefield. CGF entities may be fully autonomous (needing no human direction) or semi-autonomous (requiring some direction by a human controller who is not a participant in the virtual events). CGF entities represent friendly, opposing forces (OPFOR), and neutral battlefield participants not portrayed by manned simulators. (DIS/C) 1278.3-1996

computer graphics (A) The branch of computer science concerned with methods of creating, modifying, or analyzing pictorial data. **(B)** The use of a computer in any discipline to create, modify, or analyze images. (C) 610.6-1991

Computer Graphics Interface (CGI) (A) A computer graphics standard that provides a method for exchanging device-independent data between graphics systems or device-dependent parts of a graphics system. It is under development by the American National Standards Institute (ANSI) and the International Standards Organization (ISO). **(B)** A method for exchanging device-independent data between graphics systems or device-dependent parts of a graphics system. (C) 610.6-1991

Computer Graphics Metafile (CGM) (A) A computer graphics standard that provides a method for recording graphical information in a metafile. It was developed by the American National Standards Institute (ANSI) and the International Standards Organization (ISO). **(B)** A method for recording graphical information in a metafile. (C) 610.6-1991

computer hardware Devices capable of accepting and storing computer data, executing a systematic sequence of operations on computer data, or producing control outputs. Such devices can perform substantial interpretation, computation, communication, control, or other logical functions. (C/SE) J-STD-016-1995

computer hardware description language *See:* hardware description language.

computer input microfilm (CIM) The input to a process that converts data contained on microform into machine-readable data. (C) 610.2-1987

computer instruction A machine instruction for a specific computer. (C) [20], [85]
(2) (A) (software) A statement in a programming language, specifying an operation to be performed by a computer and the addresses or values of the associated operands; for example, Move A to B. *See also:* instruction set; instruction format. **(B) (software)** Loosely, any executable statement in a computer program. (C) 610.12-1990, 610.10-1994
(3) (A) A statement in a computer language; specifying an operation to be performed by a computer and the address or values of the associated operands; for example, MOVE A to B. *See also:* machine instruction; operation field; operand field; address field. **(B)** An instruction expressed in machine language. (C) 610.10-1994

computer instruction code A code used to represent the instruction within an instruction set. *See also:* machine code. (C) 610.10-1994w

computer instruction set The collection of computer instructions possible on a given computer. *Synonym:* machine instruction set. (C) 610.10-1994w

computer-integrated manufacturing (CIM) Use of an integrated system of computer-controlled manufacturing centers. The centers may use robotics, design automation, or CAD/CAM (computer-aided design/computer-aided manufacturing) technologies. *See also:* flexible manufacturing system. (C) 610.2-1987

computer interface equipment (1) (surge withstand capability) A device that interconnects a protective relay system to an independent computer, for example, an analog to digital converter, a scanner, a buffer amplifier. (PE/PSR) C37.90-1978s
(2) A device that interconnects a protective relay system to an independent computer, for example, a scanner or a buffer amplifier. (SWG/PE) C37.100-1992

computer interface unit A device used to connect peripheral devices with a computer. (C) 610.10-1994w

computerized axial tomography (CAT) *See:* computed tomography.

computerized healthcare information systems *See:* patient care information system.

computer language A language designed to enable humans to communicate with computers. *See also:* system profile; workload model; programming language; design language. (C) 610.12-1990
(2) (A) A language designed to enable humans to communicate with computers and computer systems. **(B)** A language that is used to control, design, or define a computer or computer program. (C) 610.13-1993, 610.10-1994

computer literacy An understanding of the capabilities, operation, and applications of computers. (C) 610.2-1987

computer-managed instruction (CMI) The use of computers for management of student progress. Activities may include record keeping, progress evaluation, and lesson assignment. *See also:* computer-based instruction. (C) 610.2-1987

computer network (1) (software) A complex consisting of two or more interconnected computers. *See also:* computer. (C/SE) [20], 729-1983s, [85]
(2) An interconnection of assemblies of computer systems, terminals and communications facilities. (LM/COM) 168-1956w
(3) A structured connection of computer systems and peripheral devices that exchange data as necessary to perform the specific function of the network. *See also:* hierarchical computer network; homogeneous computer network; heterogeneous computer network; centralized computer network; decentralized computer network; distributed computer network. (C) 610.7-1995, 610.10-1994w

computer network architecture The logical structure and the operating principles, including those concerning services, functions, nd protocols, of a computer network. *Contrast:* computer architecture. (C) 610.7-1995, 610.10-1994w

computer numerical control (CNC) Numerical control in which one or more machines that produce manufactured parts are linked together via a single computer. (C) 610.2-1987

computer operation (A) An operation which can be performed by a computer with a single instruction. **(B)** An operation performed by a functional unit within a computer. For example: an instruction fetch, or an addition. *Synonym:* machine operation. (C) 610.10-1994

computer output microfilm (COM) The end result of a process that converts and records data from a computer directly to a microform. (C) 610.2-1987

computer output microfilmer A device for producing computer output microfilm. *Synonym:* COM device. (C) 610.2-1987

uare amplitudes of volt-
and β are the phase an-
ectively, from the same
rnating voltage and the
e same primitive period
urrent; and period (prim-
ntaneous power is given

$\alpha_1 + \beta_1)]$
$3\omega t + \alpha_2 + \beta_1)]$
$3\omega t + \alpha_1 + \beta_2)]$
$\alpha_2 + \beta_2)]$

veniently as a double

$- \beta_q]$

onic component of the
nonic component of the
(harmonics)"), and E, I,
ed by the subscript. 5. If
riodic functions of the

tions of t, the instanta-

watts when the voltage
s. 7. See "reference di-
nergy will be positive if
direction and negative
n.
(Std100) 270-1966w
t which energy is deliv-
. *See also:* radio trans-
(AP/ANT) 145-1983s
) (of an electromagnetic
tantaneous electric and
of $\bar{P}(t, \bar{r})$ over a surface
power flow through the
(AP/PROP) 211-1997
l recording) A phono-
irect reproduction with-
nograph pickup.
(SP) [32]
overy time of a thermal
-energized at the instant
(EEC/REE) [87]

instantaneous relay reoperate time Reoperate time of a thermal relay measured when the heater is de-energized at the instant of contact operation. (EEC/REE) [87]

instantaneous sampling The process for obtaining a sequence of instantaneous values of a wave. *Note:* These values are called instantaneous samples. (AP/ANT) 145-1983s

instantaneous sound pressure (at a point) The total instantaneous pressure at that point minus the static pressure at that point. *Note:* The commonly used unit is the newton per square meter. (SP) [32]

instantaneous storage *See:* immediate access storage.

instantaneous suppression with automatic current regulation (thyristor) A combination of instantaneous trip or suppression and current regulation in which suppression is followed immediately by a regulated current. (IA/IPC) 428-1981w

instantaneous trip (1) (as applied to Circuit Breakers) A qualifying term indicating that no delay is purposely introduced in the tripping action of the circuit breaker. (NESC) [86] **(2)** The means to sense an overload and reduce the output current to zero, as fast as practicable. (IA/IPC) 428-1981w

instantiation (software) The process of substituting specific data, instructions, or both into a generic program unit to make it usable in a computer program. (C) 610.12-1990

instant of chopping The instant when the initial discontinuity appears. (PE/PSIM) 4-1995

instant start fluorescent lamp (illuminating engineering) A fluorescent lamp designed for starting by a high voltage without preheating of the electrodes. (EEC/IE) [126]

Institute of Electrical and Electronics Engineers (1) An organization that, among other functions, sponsors standards development. (C/BA) 14536-1995 **(2)** An international professional organization that is accredited by American National Standards Institute to develop standards for them. (C) 610.7-1995, 610.10-1994w

institutional design Emphasizes reliability, resistance to wear and use, safety to public, and special aesthetic considerations, such as the "agelessness" of the structure. (IA/PSE) 241-1990r

instruction (1) (programmable digital computer systems in safety systems of nuclear power generating stations) A meaningful expression in a computer programming language that specifies an operation to a digital computer. 554-1990 **(2) (btl interface circuits)** A binary data word shifted serially into the test logic defined by this standard in order to define its subsequent operation. (TT/C) 1149.1-1990 **(3) (software)** *See also:* computer instruction. (C) 610.12-1990 **(4)** A statement or expression consisting of an operation and its operands (if any), which can be interpreted by a computer in order to perform some function or operation. *See also:* computer instruction; microinstruction; macroinstruction. (C) 610.10-1994w

instruction address (A) The address of an instruction. **(B)** The address that must be used to fetch an instruction. (C) 610.10-1994

instruction address register An address register used to hold the address of an instruction. *Synonyms:* instruction pointer register; program register. *See also:* P register. (C) 610.10-1994w

instruction address stop An instruction address that, when it is fetched, causes execution to stop. *See also:* address stop. (C) 610.10-1994w

instructional character *See:* control character.

instructional game An instruction method employed by some computer-assisted instruction systems, in which a game is used to instruct the student on some subject. *Contrast:* simulation. (C) 610.2-1987

instructional simulation (modeling and simulation) A simulation intended to provide an opportunity for learning or to evaluate learning or educational potential; for example, a simulation in which a mock-up of an airplane cockpit is used to train student pilots. *Synonyms:* academic simulation; tutorial simulation. (C) 610.3-1989w

instruction cache A cache that stores instructions for fast access by the processor. *Contrast:* data cache. (C) 610.10-1994w

instruction code *See:* computer instruction code.

instruction control unit In a processor, the part that retrieves instructions in proper sequence, interprets each instruction, and applies the proper signals to the arithmetic and logic unit and other parts in accordance with this interpretation. *Synonym:* computer control unit. (C) 610.10-1994w

instruction counter (IC) (software) A register that indicates the location of the next computer instruction to be executed. *Synonym:* program counter. (C) 610.12-1990

instruction cycle (software) The process of fetching a computer instruction from memory and executing it. *See also:* instruction time. (C) 610.12-1990, 610.10-1994w

instruction decoder (A) The portion of the computer that determines which functions of the execution unit and the operand handler must be performed to execute the instruction. *Note:* Often implemented as part of the instruction fetch unit. **(B)** A functional component that analyzes the operation to be performed, as indicated in an instruction. *See also:* instruction processor. (C) 610.10-1994

instruction fetch unit The portion of a computer that reads the next instruction word from memory and converts the commands to the internal format used by the instruction decoder. (C) 610.10-1994w

instruction field A bit field within an instruction word. (C/MM) 1754-1994

instruction format The number and arrangement of the fields (operand, operation, and address) in a computer instruction. *See also:* address format. (C) 610.10-1994w, 610.12-1990

instruction length (software) The number of words, bytes, or bits needed to store a computer instruction. (C) 610.12-1990

instruction modifier (software) A word or part of a word used to alter a computer instruction. (C) 610.12-1990

instruction pointer register *See:* instruction address register.

instruction processor A functional component that carries out the action indicated by the instruction decoder, resulting in a possible change of machine or data state; for example, instruction decision and execution. (C) 610.10-1994w

instruction register A register that is used to hold an instruction for interpretation. (C) 610.10-1994w

instruction repertoire *See:* instruction set.

instruction set The complete set of instructions recognized by a given computer or provided by a given programming language. *Note:* In computer hardware, this term is considered to be synonymous with a computer's architecture. *Synonym:* instruction repertoire. *See also:* computer instruction set. (C) 610.10-1994w, 610.12-1990

instruction set architecture (1) (software) An abstract machine characterized by an instruction set. *See also:* abstract machine; instruction set. (C/SE) 729-1983s **(2)** An ISA defines instructions, registers, instruction and data memory, the effect of executed instructions on the registers and memory, and an algorithm for controlling instruction execution. An ISA does not define clock-cycle times, cycles per instruction, data paths, etc. This standard defines an ISA. (C/MM) 1754-1994

instruction time (1) (software) The time it takes a computer to fetch an instruction from memory and execute it. *See also:* instruction cycle. (C) 610.12-1990, 610.10-1994w **(2)** The time it takes to perform one instruction cycle. (C) 610.10-1994w

instruction trace *See:* trace.

instruction word A word that represents an instruction. *See also:* very long instruction word. (C) 610.10-1994w

instrument (1) (plutonium monitoring) A complete system designed to quantify a particular type of ionizing radiation. (NI) N317-1980r

#1
GP 2763
pah
1-18

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Attorney Docket No. **AA9-96-003**

In re application of: §
§
SOUMMYA MALLICK §
§ Examiner: **MOHAMED, A.**
Serial No.: **08/934,645** §
§
Filed: **14 APRIL 1997** §
§
For: **ADDRESS TRANSLATION BUFFER FOR DATA PROCESSING SYSTEM EMULATION MODE** § Art Unit: **2763**

RESPONSE A UNDER 37 CFR § 1.115

Assistant Commissioner of Patents
Washington, D.C. 20231

Sir:

This Response is submitted in answer to the Office Action dated October 5, 1998, having a shortened statutory period set to expire January 5, 1999.

CERTIFICATE OF MAILING
37 C.F.R. § 1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Date: 1-4-99

By: Marty Bower

REMARKS

In paragraph 3 of the present Office Action, Claims 1-4, 8, 9-13 and 16 are rejected under 35 U.S.C. § 103 as obvious over U.S. Patent No. 5,790,825 to *Traut*. That rejection is respectfully traversed, and reconsideration of the rejection is hereby requested.

With respect to exemplary Claim 1, Applicant believes that *Traut* does not render the present invention unpatentable under 35 U.S.C. § 103 because that reference fails to show or suggest each feature of Claim 1. As recited in Claim 1, the emulation of guest memory access instruction specifying a guest logical address entails "translating said guest logical address into a guest real address and thereafter translating said guest real address into a native physical address" and then "executing a semantic routine . . . utilizing said native physical address." In contrast to the invention recited in Claim 1, *Traut* does not disclose how memory access instructions are emulated, and further, does not disclose address translation. Contrary to the Examiner's assertion in paragraph 5 of the Office Action, the "translation" discussed in *Traut's* background is an emulation strategy "in which the guest instructions are analyzed and decoded only once" (col. 1, line 67 *et seq.*) and has nothing to do with address translation. Moreover, even if address translation in general were "old" as asserted by the Examiner, neither *Traut* nor knowledge generally available to those skilled in the art teaches or suggests the particular address translation recited in Claim 1. Thus, because *Traut* fails to teach or suggest the emulation of guest memory access instructions, Applicant believes that the Examiner has failed to establish a *prima facie* case of obviousness with respect to Claim 1 and respectfully request withdrawal of the rejection.

The foregoing comments made with respect to Claim 1 are also made applicable to Claims 2-4 and 8, which depend from Claim 1 and therefore necessarily incorporate all of the features of the base claim. In addition, the foregoing comments are believed to overcome the rejection of similar Claims 9-13

and 16.

In conclusion, Applicant believes that the present claims are not rendered obvious by *Traut* because that reference does not show or suggest the emulation of guest memory access instructions as recited in the present claims.

No additional fee is believed to be required; however, in the event any additional fees are required, please charge IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. However, in the event an extension of time is required, that extension of time is hereby requested. Please charge any fee associated with an extension of time to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,

Brian F. Russell

Brian F. Russell
Reg. No. 40,796
FELSMAN, BRADLEY, VADEN, GUNTER & DILLON, LLP
Suite 350 Lakewood on the Park
7600B N. Capital of Texas Hwy.
Austin, Texas 78731
(512) 343-6116

ATTORNEY FOR APPLICANTS

UNITED STA..ES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/		SHALZI	

LM71/0802

BERNARD M GOLDMAN
IBM CORPORATION
INTELLECTUAL PROPERTY LAW M S P386
522 SOUTH ROAD
POUGHKEEPSIE NY 12601-5400

EXAMINER	
FIUL, D	
ART UNIT	PAPER NUMBER
2788	3

DATE MAILED: 08/02/99

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

Notice of Allowability	Application No. 08/991,714	Applicant(s) SCALZI ET AL.
	Examiner DAN FIUL	Group Art Unit 2763

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance and Issue Fee Due or other appropriate communication will be mailed in due course.

☒ This communication is responsive to *APPLICATION FILED 12/16/97*.

☒ The allowed claim(s) is/are *1-47*.

☐ The drawings filed on ________ are acceptable.

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) ________.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: ________.

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

A SHORTENED STATUTORY PERIOD FOR RESPONSE to comply with the requirements noted below is set to EXPIRE THREE MONTHS FROM THE "DATE MAILED" of this Office action. Failure to timely comply will result in ABANDONMENT of this application. Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

☐ Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL APPLICATION, PTO-152, which discloses that the oath or declaration is deficient. A SUBSTITUTE OATH OR DECLARATION IS REQUIRED.

☒ Applicant MUST submit NEW FORMAL DRAWINGS

☐ because the originally filed drawings were declared by applicant to be informal.

☒ including changes required by the Notice of Draftsperson's Patent Drawing Review, PTO-948, attached hereto or to Paper No. 3.

☐ including changes required by the proposed drawing correction filed on ________, which has been approved by the examiner.

☐ including changes required by the attached Examiner's Amendment/Comment.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the reverse side of the drawings. The drawings should be filed as a separate paper with a transmittal lettter addressed to the Official Draftsperson.

☐ Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Any response to this letter should include, in the upper right hand corner, the APPLICATION NUMBER (SERIES CODE/SERIAL NUMBER). If applicant has received a Notice of Allowance and Issue Fee Due, the ISSUE BATCH NUMBER and DATE of the NOTICE OF ALLOWANCE should also be included.

Attachment(s)

☒ Notice of References Cited, PTO-892
☒ Information Disclosure Statement(s), PTO-1449, Paper No(s). 2
☒ Notice of Draftsperson's Patent Drawing Review, PTO-948
☐ Notice of Informal Patent Application, PTO-152
☐ Interview Summary, PTO-413
☐ Examiner's Amendment/Comment
☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
☒ Examiner's Statement of Reasons for Allowance

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER

This communication is in response to the application filed 12/16/97.

1. The following is an Examiner's statement of reasons for the indication of allowable subject matter:

2. The prior art of record teaches an emulation method/ apparatus executing an incompatible computer program on a target processor comprising storing in a target system memory routines for each operation code where each target routine performing a function meant to emulate the function of the operational code. Although an emulation method/apparatus as described above has been cited, the prior art of record does not teach associating one or more patching instructions with a target instruction in a target routine when the target instruction requires modification for enabling the target routine to provide execution results identically to execution results defined for the corresponding incompatible instruction by the incompatible architecture and not associating any patching instruction with any target not requiring modification. For these reasons claims 1 and 7 are deemed to be allowable over the prior art of record and claims 2-6 and 8-47 are allowable by dependency.

Serial Number: 08/991,714

Art Unit: 2763

3. Any inquiry concerning this communication or earlier communication from the examiner should be directed to Dan Fiul, telephone number (703)305-3853.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist, telephone number (703)305-3900.

A shortened statutory period for response to this action is set to expire 3 (three) months and 0 (zero) days from the date of this letter. Failure to respond within the period for response will cause the application to become abandoned (see MPEP 710.02, 710.02(b)).

Dan Fiul D.F.
July 26, 1999

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER

FORM PTO-892 (REV. 03-78)

U.S. DEPA[illegible]ENT OF COMMERCE PATENT AND TRADEMARK OFFICE

NOTICE OF REFERENCES CITED

SERIAL NO.	G[illegible]P ART UNIT	ATTACHMENT TO PAPER NO	
08991,714	2763		3

APPLICANT(S): SCALZI ET AL.

U.S. PATENT DOCUMENTS

*		DOCUMENT NO	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	A	5471612	11/28/95	SCHLAFLY	707	104	
	B	5574927	11/12/96	SCANTLIN	395	500 44	
	C	5751982	5/12/98	MORLEY	712	209	
	D	5857094	1/5/99	NEMIROVSKY	395	500.49	
	E	5896522	4/20/99	WARD ET AL.	395	500.44	
	F	5909567	6/1/99	NOVAK ET AL.	712	208	
	G						
	H						
	I						
	J						

FOREIGN PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
	L						
	M						
	N						
	O						
	P						
	Q						

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, Etc.)

	R	ANNEXSTEIN ET AL., ACHEIVING MULTILANGUAGE BEHAVIOR IN BIT-SERIAL SIMD ARCHITECTURES VIA EMULATION, 2/90, PAGES 186-195
	S	HOOKWAY, DIGITAL FX!32 RUNNING 32-BIT X86 APPLICATIONS ON ALPHA NT 3/97, PAGES 37-42
	T	

EXAMINER	DATE	
DAN FIUL	July 26, 1999	

*** A copy of this reference is not being furnished with this office action. (See Manual of Patent Examining Procedure, section 707.05(a).)**

FORM PTO 948 (REV. 11-97) U.S. DEPARTMENT OF COMMERCE-Patent and Trademark Office Application No. 991714

NOTICE OF DRAFTPERSON'S PATENT DRAWING REVIEW

The drawing filled (insert date) 12/16/97 are:

A. _____ not objected to by the Draftperson under 37 CFR 1.84 or 1.152.

B. X objected to by the Draftperson under 37 CFR 1.84 or 1.152 as indicated below. The Examiner will require submission of new, corrected drawings whe necessary. Corrected drawings must be submitted according to the instructions on the back of this notice.

1. DRAWINGS. 37 CFR 1.84(a): Acceptable categories of drawings: Black ink. Color.
   - _____ Color drawing are not acceptable until petition is granted. Fig.(s) _____
   - _____ Pencil and non black ink is not permitted. Fig(s) _____
2. PHOTOGRAPHS. 37 CFR 1.84(b)
   - _____ Photographs are not acceptable until petition is granted,
   - _____ 3 full-tone sets are required. Fig(s) _____
   - _____ Photographs not properly mounted (must brystol board or photographic double-weight paper). Fig(s) _____
   - _____ Poor quality (half-tone). Fig(s) _____
3. TYPE OF PAPER. 37 CFR 1.84(e)
   - _____ Paper not flexible, strong, white and durable. Fig.(s) _____
   - _____ Erasures, alterations, overwritings, interlineations, folds, copy machine marks not acceptable. (too thin)
   - _____ Mylar, vellum paper is not acceptable (too thin). Fig(s) _____
4. SIZE OF PAPER. 37 CFR 1.84(f): Acceptable sizes:
   - _____ 21.0 cm by 29.7 cm (DIN size A4)
   - _____ 21.6 cm by 27.9 cm (8 1/2 x 11 inches)
   - _____ All drawings sheets not the same size. Sheet(s) _____
5. MARGINS. 37 CFR 18.4(g): Acceptable margins:
   - Top 2.5 cm Left 2.5 cm Right 1.5 cm Bottom 1.0 cm SIZE: A4 Size
   - Top 2.5 cm Left 2.5 cm Right 1.5 cm Bottom 1.0 cm SIZE: 8 1/2 x 11
   - X Margins not acceptable. Fig(s) 2-5, 10
   - X Top (T) X Left (L)
   - _____ Right (R) _____ Bottom (B)
6. VIEWS. CFR 1.84(h)
   REMINDER: Specification may require revision to correspond to drawing changes.
   - _____ Views connected by projection lines or lead lines. Fig.(s) _____

   Partial views. 37 CFR 1.84(h)(2)
   - _____ Brackets needed to show figure as one entity. Fig.(s) _____
   - _____ Views not labeled separately or properly. Fig.(s) _____
   - _____ Enlarged view not labeled separately or properly. Fig.(s) _____
7. SECTIONAL VIEWS. 37 CFR 1.84(h)(3)
   - _____ Hatching not indicated for sectional portions of an object. Fig.(s) _____
   - _____ Sectional designation should be noted with Arabic or Roman numbers. Fig.(s) _____
8. ARRANGEMENT OF VIEWS. 37 CFR 1.84(i)
   - _____ Words do not appear on a horizontal, left-to-right fashion when page is either upright or turned, so that the top becomes the right side, except for graphs. Fig.(s) _____
   - _____ Views not on the same plane on drawing sheet. Fig.(s) _____
9. SCALE. 37 CFR 1.84(k)
   - _____ Scale not large enough to show mechansim without crowding when drawing is reduced in size to two-thirds in reproduction. Fig.(s) _____
10. CHARACTER OF LINES, NUMBERS, & LETTERS. 37 CFR 1.84(l)
    - _____ Lines, numbers & letters not uniformly thick and well defined, clean, durable and black (poor line quality). Fig.(s) _____
11. SHADING. 37 CFR 1.84(m)
    - _____ Solid black areas pale. Fig.(s) _____
    - _____ Solid black shading not permitted. Fig.(s) _____
    - _____ Shade lines, pale, rough and blurred. Fig.(s)
12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1.48(p)
    - _____ Numbers and reference characters not plain and legible. Fig.(s) _____
    - _____ Figure legends are poor. Fig.(s) _____
    - _____ Numbers and reference characters not oriented in the same direction as the view. 37 CFR 1.84(p)(3) Fig.(s) _____
    - X Engligh alphabet not used. 37 CFR 1.84(p)(3) Fig.(s) _____
    - X Numbers, letters and reference characters must be at least .32 cm (1/8 inch) in height. 37 CFR 1.84(p)(3) Fig.(s) 10
13. LEAD LINES. 37 CFR 1.84(q)
    - _____ Lead lines cross each other. Fig.(s) _____
    - _____ Lead lines missing. Fig.(s) _____
14. NUMBERING OF SHEETS OF DRAWINGS. 37 CFR 1.48(t)
    - _____ Sheets not numbered consecutively, and in Ababic numerals beginning with number 1. Fig.(s) _____
15. NUMBERING OF VIEWS. 37 CFR 1.84(u)
    - _____ Views not numbered consecutively, and in Abrabic numerals, beginning with number 1. Fig.(s) _____
16. CORRECTIONS. 37 CFR 1.84(w)
    - _____ Corrections not made from PTO-948 dated _____
17. DESIGN DRAWINGS. 37 CFR 1.152
    - _____ Surface shading shown not appropriate. Fig.(s) _____
    - _____ Solid black shading not used for color contrast. Fig.(s) _____

COMMENTS

REVIEWER Q.D DATE 4/2/98 TELEPHONE NO. 7033058404

ATTACHMENT TO PAPER NO. _____

PTO COPY

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

NOTICE OF ALLOWANCE AND ISSUE FEE DUE

LM71/0802

BERNARD M GOLDMAN
IBM CORPORATION
INTELLECTUAL PROPERTY LAW M S P386
522 SOUTH ROAD
POUGHKEEPSIE NY 12601-5400

APPLICATION NO.	FILING DATE	TOTAL CLAIMS	EXAMINER AND GROUP ART UNIT		DATE MAILED
08/991,714	12/16/97	047	FIUL, D	2763	08/02/99

First Named Applicant: SCALZI,   35 USC 154(b) term ext. = 0 Days.

TITLE OF INVENTION: PREPROCESSING OF STORED TARGET ROUTINES FOR EMULATING INCOMPATIBLE INSTRUCTIONS ON A TARGET PROCESSOR

ATTY'S DOCKET NO.	CLASS-SUBCLASS	BATCH NO.	APPLN. TYPE	SMALL ENTITY	FEE DUE	DATE DUE
0 PO996138	395-500.470	W53	UTILITY	NO	$1210.00	11/02/99

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. <u>PROSECUTION ON THE MERITS IS CLOSED.</u>

THE ISSUE FEE MUST BE PAID WITHIN <u>THREE MONTHS</u> FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. <u>THIS STATUTORY PERIOD CANNOT BE EXTENDED.</u>

HOW TO RESPOND TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.
If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is changed, pay twice the amount of the FEE DUE shown above and notify the Patent and Trademark Office of the change in status, or

B. If the status is the same, pay the FEE DUE shown above.

If the SMALL ENTITY is shown as NO:

A. Pay FEE DUE shown above, or

B. File verified statement of Small Entity Status before, or with, payment of 1/2 the FEE DUE shown above.

II. Part B-Issue Fee Transmittal should be completed and returned to the Patent and Trademark Office (PTO) with your ISSUE FEE. Even if the ISSUE FEE has already been paid by charge to deposit account, Part B Issue Fee Transmittal should be completed and returned. If you are charging the ISSUE FEE to your deposit account, section "4b" of Part B-Issue Fee Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give application number and batch number.
Please direct all communications prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PATENT AND TRADEMARK OFFICE COPY

PTOL-85 (REV. 10-96) Approved for use through 06/30/99. (0651-0033)

*U.S. GPO. 1998-437-639/80023

2396
#6
KW
1-3-97

PATENT

Docket No. 1963-4185

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s)	:Schwarz et al.	
Serial No.	:08/414,250	Group Art Unit:2306
Filed	:March 31, 1995	Examiner:C. Ngo
For	:IMPLEMENTATION OF BINARY FLOATING POINT USING HEXADECIMAL FLOATING POINT UNIT	

PETITION AND FEE FOR EXTENSION OF TIME (37 C.F.R. §1.136(a))

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Sir:

1. This is a petition for an extension of time for filing Amendment Under 37 CFR §1.111 .

2. The communication in connection with the matter for which this extension is requested

   [X] is filed herewith.

   [ ] has been filed on ______________________________

3. [ ] Applicant is a small-entity -- verified statement is attached [ ], or has already been filed. [ ].

4.

		Total Months Requested	Fee for Other than Small Entity	Fee for Small Entity
a.	[ ]	one month	$110.00	$55.00
b.	[ ]	two months	$390.00	$195.00
c.	[X]	three months	$930.00	$465.00
d.	[ ]	four months	$1,470.00	$735.00

   e. [ ] An extension for _____ months has already been secured for filing the above-identified communication and the fee paid therefor of $ __________ is deducted from the total fee due for the total months of extension now requested. The fee for this extension ($ _______), minus the fee previously paid ($ __________) equals $ ___________ (total fee due).

5. [ ] A check in the amount of $________________ to cover the extension fee is attached.

6. [X] Charge fee to Deposit Account No. 13-4500. Order No. 1963-4185 . A DUPLICATE COPY OF THIS SHEET IS ATTACHED.

262429_1

7. [X] The Assistant Commissioner is hereby authorized to charge any additional fees which may be required by this paper, or credit any overpayment to Deposit Account No. 13-4500. Order No. 1963-4185. A DUPLICATE COPY OF THIS SHEET IS ATTACHED.

Respectfully submitted,

MORGAN & FINNEGAN, L.L.P.

Dated: December 11, 1996

By: ______________________

David V. Rossi

Registration No. 36,659

Mailing Address:

MORGAN & FINNEGAN, L.L.P.
345 Park Avenue
New York, New York 10154
(212) 758-4800
(212) 751-6849 Telecopier

FORM: PET-XOT.NY
Rev. 10/1/96

#7/a
KW
1-3-97

Docket No. PO994050 **Patent**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant	**:**	Schwarz et al.
Serial No	**:**	08/414,250
Filed	**:**	March 31, 1995
Group Art Unit	**:**	2306
Examiner	**:**	NGO, C.
For	**:**	***IMPLEMENTATION OF BINARY FLOATING POINT USING HEXADECIMAL FLOATING POINT UNIT***

AMENDMENT UNDER 37 C.F.R. §1.111

RECEIVED
JAN 02 1997
GROUP 2300

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In response to the Office Action dated June 11, 1996, please amend the application as follows:

IN THE CLAIMS:

Please amend the claims as follows:

a1

1. (Amended) A computer system supporting a plurality of floating point architectures, each floating point architecture having at least one format, said system comprising:

a floating point unit having an internal dataflow [adapted to accommodate] according to an internal floating point format having a number of exponent bits which is the minimum number required to support each of said plurality of floating point

1-14

architectures, said floating point unit performing floating point operations in said internal floating point format; and

a1

conversion means for converting between each one of said plurality of floating point architectures and said internal [dataflow] floating point format such that an operand of any one of said plurality of floating point architectures input to said floating point unit is converted into said internal floating point format for operation by said floating point unit, and the result of the operation is converted into any one of said plurality of floating point architectures.

2. (Amended) The [A] computer system according to claim 1, wherein said plurality of floating point architectures includes [supporting] a hexadecimal floating point architecture and IEEE 754 binary floating point architecture[, said system comprising:

a floating point unit having an internal dataflow adapted to accommodate said hexadecimal floating point architecture and said IEEE 754 plurality of floating point architectures; and

conversion means for converting between each one of said plurality of floating point architectures and said internal dataflow].

4. (New) The computer system according to claim 2, wherein said hexadecimal floating point architecture is IBM S/390 hexadecimal architecture.

a2

45 282412.1

5. (New) The computer system according to claim 4, wherein said internal floating point format is a hexadecimal format that has a fourteen bit exponent biased by 8192, a sign bit, and a 56 bit fraction.

a2

6. (New) The computer system according to claim 1, wherein said plurality of floating point architectures includes a binary architected format and an hexadecimal architected format, and said internal format is a hexadecimal internal format.

7. (New) The computer system according to claim 6, wherein said hexadecimal architected format and said hexadecimal internal format each have a common predetermined bias type, said predetermined bias type being either an even bias or an odd bias.

8. (New) The computer system according to claim 7, wherein said predetermined bias type is even, and said hexadecimal internal format has a nonzero positive integer number (N+1) of exponent bits and a bias equal to 2^N.

9. (New) The computer system according to claim 1, wherein said plurality of floating point architectures includes a binary architected format and an hexadecimal architected format, and said internal format is a hexadecimal internal format, and wherein said conversion means includes:

a first converter that converts said hexadecimal architected format to said hexadecimal internal format;

a second converter that converts said binary architected format to said hexadecimal internal format;

a third converter that converts said hexadecimal internal format into said binary architected format; and

a2

a fourth converter that converts said hexadecimal internal format into said hexadecimal architected format.

10. (New) The computer system according to claim 1, wherein said plurality of floating point architectures includes a binary architected format and an hexadecimal architected format, and said internal format is a hexadecimal internal format, and wherein said conversion means includes:

a first converter that applies a transformation to convert said hexadecimal architected format to said hexadecimal internal format, and applies said transformation to binary architected format data to provide transformed binary architected format data;

a second converter that converts said transformed binary architected format data to said hexadecimal internal format;

a third converter that converts said hexadecimal internal format into said transformed binary architected format data; and

a fourth converter that applies a second tranformation that converts said hexadecimal internal format into said hexadecimal architected format, and applies said second transformation to said transformed binary architected format data to provide said binary architected format data.

11. (New) In a computer system, a floating point unit that supports a first floating point architecture and a second floating point architecture, said first and second floating point architectures each having at least one format, said system comprising:

a floating point unit having an internal dataflow with an internal floating point format which supports both said first floating point architecture and said second floating point architecture, and that performs floating point operations on data formatted in said internal floating point format;

a2

a first converter that applies a transformation to convert an operand of said first floating point architecture type to said internal floating point format, and applies said transformation to an operand of said second floating point architecture type to provide transformed second floating point architecture type data;

a second converter that converts said transformed second floating point architecture type data to said internal floating point format;

a third converter that converts data of said internal floating point format into data of said transformed second floating point architecture type; and

a fourth converter that applies a second tranformation that converts data of said internal floating point format into data of said first floating point architecture, and applies said second transformation to data of said transformed second floating point architecture type to provide data of said second floating point architecture type.

12. (New) The computer system according to claim 11, wherein an operand of said first floating point architecture type is directly converted into said internal floating point

format by said first converter for operation by said floating point unit, and wherein an operand of said second floating point architecture type is converted into said internal floating point format by said first converter and said second converter operating in series.

13. (New) The computer system according to claim 11, wherein data in said internal floating point format is directly converted into data of said first floating point architecture by said fourth converter, and wherein data in said internal floating point format is converted into data of said second floating point architecture type by said third converter and said fourth converter operating in series.

a2

14. (New) The computer system according to claim 11, wherein said internal floating point format has a minimum nonzero positive number of exponent bits, N, to support both first and second floating point architectures.

15. (New) The computer system according to claim 11, wherein said first and second floating point architectures have different bias types, the bias types being either of even type or odd type, said internal format having a bias type that matches said first floating point architecture.

16. (New) The computer system according to claim 15, wherein said internal floating point format has a minimum nonzero positive number of exponent bits, N, to support both first and second floating point architectures, and wherein said internal floating point has an even bias of $2^{(N-1)}$.

17. (New) The computer system according to claim 11, wherein said internal format has the same base and bias type as one of said first and second floating point architectures, the same bias type being either even or odd.

a^2

REMARKS

I. Introduction

This Amendment under 37 C.F.R. §1.111 is submitted in response to the outstanding Office Action dated June 11, 1996, and is accompanied by a Petition for Extension of Time and fee. Claims 1-17 are now pending in the application. Claims 1 and 2 have been amended to clarify that the computer system supports a *plurality* of floating point *architectures, each having at least one format*, by converting data of any of these *architectures* into a common internal floating point *format* in which floating point operations are performed. That is, Applicant has clarified floating point architectures with respect to different formats (i.e., precision lengths) of the same floating point architecture. New claims 4-17 have been entered. Applicants respectfully request reconsideration in view of the herewith presented amendments and remarks.

II. Rejection Under 35 U.S.C. §112, Second Paragraph

The specification is objected to under 35 U.S.C. §112, first paragraph, as failing to adequately teach how to make and/or use the invention, i.e. failing to provide an enabling disclosure. Claims 1-3 are rejected based on this objection.

In particular, the Examiner states that:

SC

> Applicant fails to clearly disclose how binary floating point data in the "Architected FPRs" (10) are processed by the floating point unit. It is noted that the inputs and outputs of the floating point unit which are the inputs to the MUXs (25) and (26) connected to "Hex-I to Hex-a" (11) and (12), and the output from the "Hex-I to Hex-a" (24) as illustrated in Fig. 1, all must have been in hex floating point format.

Applicants respectfully traverse the objection and the rejection, and submit that the specification enables an ordinarily skilled artisan to understand how the floating point unit processes binary floating point data stored in the Architected FPRs.

Applicants note that, based on the Examiner's comments, it appears that the names of the format converters may be the reason for the Examiner's confusion as to the operation of these converters with respect to binary data. As noted by the Examiner, all architected data that is loaded into the floating point execution unit must pass through the Hex-A to Hex-I converters (11) and (12), and all data stored from the unit must pass through the Hex-I to Hex-A (24) converter.

Note that the internal format is designed to be almost identical to the hex architected format and thus these two conversion processes minimally alter the data. The only difference between hex architected format and hex internal format is the use of a 7 bit exponent versus a 14 bit exponent, respectively. The transformation of hex architected form to hex internal involves a process similar to sign extending an operand.

$$Internal\_Exponent(0:13) \leftarrow Architected\_Bits(1,\overline{1},\overline{1},\overline{1},\overline{1},\overline{1},\overline{1},\overline{1},2:7)$$

where $\overline{1}$ indicates the concatenation of the complement of bit 1. This simple transformation is applied to all data loaded into the unit--regardless of whether the data is in binary floating point or hexadecimal floating point. Note that this process is easily reversed by concatenating bits 0 and 8 to 13 of the internal exponent.

Architected_Bits(1 : 7) ← Internal_Exponent(0, 8 : 13)

Also note that, in accordance with such an embodiment, the hexadecimal fraction is not altered and only the exponent is affected. This feature is considered in the binary format converters.

Basically, then, as described according to the embodiment disclosed in the specification, for each of the Bin-A to Hex-I converters (15) (16), the processing of the data through the Hex-A to Hex-I converters is reversed to recreate the binary architected format, and then it is converted to Hex internal format. Simply stated, in the Hex-A to Hex-I converter binary data undergoes the same transformation as hexademical data. For hexadecimal data, this simple transformation results in the hexadecimal internal format which is provided to the Hex Internal Dataflow. For binary data, this same transformation produces an intermediate data state which is easily accounted for (e.g., reversed) in the subsequent Bin-A to Hex-I converter process which in turn provides to the Hex Internal Dataflow the hexadecimal internal format equivalent of the binary formatted data.

Similarly, the Hex-I to Hex-A converter (24) performs the simple reversal of the Hex-A to Hex-I converter transformation, and the Hex-I to Bin-A converter (21) produces an intermediate state that when processed by Hex-I to Hex-A converter (24) produces the architected binary format.

As stated above, the Examiner's confusion may result from the name of the converters. If Applicants had relabeled the converters A, B, C, and D, for Hex-A to Hex-I, Bin-A to Hex-I, Hex-I to Bin-A, and Hex-I to Hex-A, respectively, then it could be stated that: converting from architected hexadecimal format to hexadecimal internal requires processing by the A converter; converting from architected binary format to hexadecimal

internal requires processing by both the A and B converters; converting from hexadecimal internal to architected hexadecimal format requires the D converter; and converting from hexadecimal internal to architected binary format requires the C and D converters.

As an aside, and as further discussed hereinbelow in connection with the rejections under 35 U.S.C. §§102 and 103, it may be appreciated that a novel feature of Applicant's invention is that instead of using four *distinct* converters to directly transform between (i.e., to/from) architected hexadecimal format and hexadecimal internal (i.e. two converters) and between architected binary format and hexadecimal internal (i.e., two converters), in accordance with a feature of a preferred embodiment of the invention, in order to optimize for architected hexadecimal format the hex-int to/from hex-arch converters perform a simple transformation and are part of the data flow for all conversions (i.e., for both architected binary format to/from hex internal, and architected hexadecimal to/from hex internal).

Applicants respectfully submit that the handling of the binary floating point format is clearly described throughout the specification, including the detailed operational descriptions of each converter. For instance, page 14, line 4, shows the equation which undoes the hex-arch to hex-int conversion as the first step in the bin-arch to hex-int converter. The simplicity of the hex-arch to hex-int converter is described on page 7, line 17 to page 8, line 3 and also on page 10, line 19 to page 11, line 16. And a specific example for binary addition is given on pages 30 to 36. Page 30, line 19 to page 31, line 3 describes the undoing of hex-arch to hex-int conversion.

Applicants also respectfully submit that the conversion from hex-int to bin-arch is clearly described on page 24 where only 7 bits of exponent are actually driven to the exponent register, and on page 27, last paragraph where the remaining least significant bits

of exponent are driven to the fraction register. In the binary addition example, page 35 lines 7 to 13 describe dealing with output conversion.

In view of the foregoing, Applicants respectfully submit that the specification is clearly enables an ordinarily skilled artisan to practice the invention, including to understand how the floating point unit processes binary floating point data stored in the Architected FPRs. Accordingly, Applicants respectfully request that the objection to the specification, and the rejection of the claims, under 35 USC §112, first paragraph, be withdrawn.

III. Rejection Under 35 U.S.C. §112, Second Paragraph

Claims 1-3 are rejected under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. The Examiner stated that the phrase "adapted to accommodate" in claim 1 is indefinite, and also stated that in claim 2 "IEEE 754" is indefinite because the format of "IEEE 754" varies by times.

Applicants have amended claim 1, and respectfully submit that the rejection has been obviated. With respect to "IEEE 754", Applicants respectfully traverse this rejection.

Applicants respectfully submit that this term is not indefinite to the extent that it distinctly and reasonably conveys to an ordinarily skilled artisan a particular binary floating point format. Also, Applicants respectfully submit that the claims need not define every present and future possible embodiment, and need not delimit only operative embodiments to the exclusion of inoperative embodiments, present or future. Further, the term "IEEE 754" has no more varied meaning than a multitude of language terms normally used in claims and which necessarily may have meaning which changes with time and development of the art.

Moreover, the term "IEEE 754" defines a binary floating point format which has arguably a more specific and distinct meaning than, for instance, simply "binary floating point format".

In view of these amendments and remarks, Applicants respectfully submit that the rejection under 35 U.S.C. §112 has been obviated. Applicants, therefore, respectfully request the withdrawal of the rejection of the claims under 35 U.S.C. §112, second paragraph.

IV. <u>The 35 U.S.C. §§102 and 103 Rejections</u>

Claim 1 is rejected under 35 U.S.C §102(b) as being anticipated by Saini, U.S. Patent 4,949,291 ("Saini"). Claims 2 and 3 are rejected under 35 U.S.C §103 as being unpatentable over Saini.

Claim 1 is also rejected under 35 U.S.C §102(e) as being anticipated by Naini, U.S. Patent 5,523,961 ("Naini"). Further, Claims 2 and 3 are rejected under 35 U.S.C §103 as being unpatentable over Naini.

Applicants respectfully traverse these rejections, and submit that, for the reasons presented below, Saini (Naini) neither anticipates nor renders obvious Applicants' amended claims. Since there are many common distinctions of Applicants' claimed invention over Saini and Naini individually, Applicant hereinbelow discusses both Saini and Naini concurrently, but individually, with respect to these distinctions.

As may be appreciated, Applicants invention is directed to a computer system supporting a *plurality of floating point architectures*. The computer system includes a floating point unit having an internal dataflow according to an internal floating point format having a number of exponent bits which is the minimum number required to support each of

the plurality of floating point architectures. Floating point operations are performed in the internal floating point format. There are conversion means for converting to/from each one of the plurality of floating point architectures and the internal floating point format.

A feature of a preferred implementation of Applicants' invention is that conversion between data of a first floating point architecture type to/from the internal format, in addition to a separate set of converters, also employs the converters exclusively required for conversion between data of the second floating point architecture type to/from the internal format. This feature is practicable because the second floating point architecture type is optimized for easy conversion to/from the internal format, and thus the effect of these conversions may easily be reversed when applied during converting to/from data of the first floating point architecture type.

In a preferred embodiment of the invention, a hexadecimal internal format is optimized for a hex architected format (e.g., IBM S/390 architecture), and also supports a binary architected format (e.g., IEEE 754 Binary Floating Point standard). In the preferred implementation, a hex internal format with an biasing of the form 2^N (i.e., even biasing corresponding to S/390; $2^N = 8192$) is a key feature to providing a high performance hexadecimal based architected format (i.e., S/390).

In sharp contrast to Applicants claimed invention, Saini (Naini) neither teaches nor suggests a computer system with "a floating point unit having an internal dataflow according to an internal floating point format having a number of exponent bits which . . . [supports] each of [a] . . . *plurality of floating point architectures*". (See e.g., claim 1; emphasis added).

Saini describes representing IEEE single, double and double extended formats (i.e., different precision lengths of the *same floating point architecture*) using an internal binary format with a 17 bit binary exponent. Saini converts to the internal format by adding a constant to the architected exponent. Naini involves a similar conversion process between IEEE (architected) formats and an internal binary exponent format but uses a sign-extension method to perform the conversion. In Saini (Naini), the internal format has a binary exponent correponding to the binary architecture, and there is no description or suggestion of supporting an external hexadecimal exponent format. Only one architecture is discussed, which is IEEE 754 standard. Saini (Naini), therefore, only teaches implementing only one architecture which has multiple formats. As stated, Applicants' claimed invention supports a plurality of *architectures* (e.g., hex and binary architectures), which Saini (Naini) does not teach or suggest.

Applicants, therefore, respectfully submit that Saini (Naini) does not anticipate Applicants' invention as claimed in claim 1. Accordingly, Applicants respectfully submit that claims 2 and 3 should be allowed as being dependent on allowable claim 1. In addition, however, Applicants further emphasize that Saini (Naini) does not suggest Applicants claimed invention.

First, Applicants emphasize that Saini (Naini) does not teach or suggest a plurality of external *floating point architectures*, and *a fortiori*, further does not teach or suggest a common internal format supporting a plurality of external floating point architectures. Accordingly, even assuming *arguendo* that it would be obvious to modify Saini (Naini) to substitute an internal floating point format having a base (e.g., hex format) different from the external architecture (e.g., binary architecture), Applicants' claimed invention could not

result because this presumed modified system would lack two distinct external *floating point architectures*. Applicant further submits, however, that Saini (Naini) teach away from having an internal format that has a different base than the external architecture to the extent that Saini (Naini) are directed to rapidly and efficiently converting between external and internal formats, and having different internal and external bases would be inconsistent with this objective. This distinction of Applicants' claimed invention over a presumed modified system according to Saini (Naini) further highlights the inappropriateness and inadequacy of Saini (Naini) with respect to Applicants' claimed invention.

In addition, as noted, Applicants submit that Saini and Naini are each focussed on the problem of efficiently converting floating point data of different formats (i.e., different precision lengths, but the same architecture) into a common format. This problem is in no way directed to the problem of providing a floating point unit that accommodates two distinct floating point architectures (e.g., hexadecimal and binary architectures).

That is, Applicant respectfully submits that converting between a common internal format and multiple formats of the *same architecture* (e.g., IEEE Binary Floating Point) is starkly different from converting between a common internal format and a plurality of *different architected floating point types*. This distinction may be appreciated by considering the distinctions between IBM S/390 hexadecimal floating point and IEEE 754 Binary floating point: each of these architectures inherently requires and prescribes *distinct floating point dataflows*. It is appreciated that converting between architected types has a more significant impact on hardware (and concomitantly chip real estate) requirements and further, requires consideration of performance/hardware tradeoffs/optimization with respect to each architected type. Thus, the problem of accommodating distinct floating point *architectures* on a single

computing system is vastly different than accomodating multiple formats (i.e., precision lengths) of a single floating point architecture.

Simply stated, Saini (Naini) is inapposite: not only is Applicants' claimed invention neither taught nor suggested by Saini (Naini) but also, Applicants' invention is directed to a problem distinct from that addressed by Saini (Naini). There is no suggestion in Saini (Naini) of supporting a plurality of architectures. Nevertheless, even assuming *arguendo* that the different external formats of Saini (Naini) were modified to be different floating point architectures, Applicants' claimed invention could not result because there is no teaching or suggestion of how to define the internal format relative to the architectures, or how to optimize floating point operation (e.g., including conversion) with respect to a plurality of architectures.

In view of the foregoing, Applicants respectfully submit that Saini (Naini) neither teaches nor suggests a system for supporting a plurality of floating point architectures as claimed by Applicants. Therefore, Applicants respectfully request that the rejections under 35 U.S.C. §§102(b), 102(e), and 103 be withdrawn, and submit that claims 1-3 should be allowed.

V. New Claims

Applicants have entered new claims 4-17 to further claim Applicants' invention. No new matter has been added. Applicants respectfully submit that these new claims are allowable over the prior art of record based on the hereinabove remarks.

In addition, Applicants respectfully note that claims 10 and 11 recite additional patentably distinct subject matter, including the use of a "single stage" conversion of a first

floating point architecture to/from the internal format, and the use of a "double stage" conversion--one of the stages overlapping the "single stage" conversion--of a second floating point architecture to/from the internal format.

VI. Conclusion

In view of the above amendments and remarks, Applicants respectfully submit that the application is in condition for allowance. Reconsideration and withdrawal of the Examiner's rejections is respectfully requested and allowance of all pending claims is respectfully submitted.

If any outstanding issues remain, or if the Examiner has any suggestions for expediting allowance of this application, the Examiner is invited to contact the undersigned at the telephone number below.

Early and favorable action is earnestly solicited.

Respectfully submitted,

MORGAN & FINNEGAN

Date: December 11, 1996

By: David V. Rossi
Registration No. 36,659

MAILING ADDRESS:
Morgan & Finnegan
345 Park Avenue
New York, New York 10154
(212) 758-4800

PATENT

Docket No. PO994050

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s)	:Schwarz et al.	Group Art Unit:2306	
Serial No.	:08/414,250	Examiner:C. Ngo	
Filed	:March 31, 1995		
For	:IMPLEMENTATION OF BINARY FLOATING POINT USING HEXADECIMAL FLOATING POINT UNIT		

AMENDMENT FEE TRANSMITTAL

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Sir:

Transmitted herewith is an Amendment for the above-identified application.

[X] No additional fee is required.

[ ] The additional fee has been calculated as shown below:

CLAIMS AS AMENDED

	Claims Remaining After Amendment		Highest No. Covered by Previous Payments	Present Extra	Rate	Additional Fee
Total Claims*		-		=	x $22.00	$_____
Independent Claims		-		=	x $80.00	$_______
Multiple Dependent Claim(s)	(If claims added by amendment include Multiple Dependent Claim(s) and there was no Multiple Dependent Claim(s) in					$_________

* Includes all independent and single dependent claims and all claims referred to in multiple dependent claims. See 37 C.F.R. § 1.75(c).

262427_1

Docket No. P0994050

application before amendment add
$260.00 to additional fee.) Total: $ -0-

[ ] Verified Statement of "Small Entity" Status Under 37 CFR § 1.27 filed ______________. Reduced Fees Under 37 CFR § 1.9(f) (50% of total) paid herewith. $________

[ ] Charge fee to Deposit Account No. 09-0463. Order No. PO994050.
A DUPLICATE COPY OF THIS SHEET IS ATTACHED.

[X] The Assistant Commissioner is hereby authorized to charge any additional fees which may be required for this amendment, or credit any overpayment to Deposit Account No. 09-0463. Order No. PO994050.
A DUPLICATE COPY OF THIS SHEET IS ATTACHED.

[ ] ____ Page(s) of substitute Sequence Listing

[ ] ____ Computer disk(s) containing substitute Sequence Listing

[ ] Statement under 37 C.F.R. § 1.825(b) that the computer and paper copies of the substitute Sequence Listing are the same.

[ ] A check in the amount of $ ____________ to cover the filing fee is attached.

Respectfully submitted,

MORGAN & FINNEGAN, L.L.P.

Dated: December 11, 1996

By: ______________________
David V. Rossi
Registration No. 36,659

Mailing Address:

MORGAN & FINNEGAN, L.L.P.
345 Park Avenue
New York, New York 10154
(212) 758-4800
FORMS1-AMD-Tele-NY
Rev. 10/1/96

262427_1

MAIL ROOM
DEC 11 1996
63
PAT. & TRADEMARK OFC.

PATENT
Docket No. PO994050

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants	:	Schwarz, et al.	Group Art Unit: 2306
Serial No.	:	08/414,250	Examiner: C. Ngo
Filed	:	March 31, 1995	
For	:	IMPLEMENTATION OF BINARY FLOATING POINT USING HEXADECIMAL FLOATING POINT UNIT	

EXPRESS MAIL CERTIFICATE

Express Mail Label No. EH742649100 US

Date of Deposit December 11, 1996

I hereby certify that the following attached paper(s) or fee

RECEIVED
JAN 02 1997
GROUP 2300

Amendment Under 37 CFR §1.111
Amendment Fee Transmittal
Petition For Extension of Time
Return Postcard

are being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. §1.10 on the date indicated above and are addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

David V. Rossi
(Typed or printed name of person mailing paper(s) or fee)

[signature]
(Signature of person mailing paper(s) or fee)

Mailing Address:

MORGAN & FINNEGAN, L.L.P.
345 Park Avenue
New York, New York 10154
(212) 758-4800
(212) 751-6849 Telecopier

FORM: EXP-MAIL.NY
Rev. 3/27/95

262431_1

COMPREHENSIVE DICTIONARY OF ELECTRICAL ENGINEERING

EDITOR-IN-CHIEF
Phillip A. Laplante

CRC Taylor & Francis
Taylor & Francis Group

Published in 2005 by
CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-10: 0-8493-3086-6 (Hardcover)
International Standard Book Number-13: 978-0-8493-3086-5 (Hardcover)
Library of Congress Card Number 2004058572

Library of Congress Cataloging-in-Publication Data

Comprehensive dictionary of electrical engineering / editor-in-chief Phillip A. Laplante.-- 2nd ed.
p. cm.
ISBN 0-8493-3086-6 (alk. paper)
1. Electric engineering--Dictionaries. I. Title: Electrical engineering. II. Laplante, Phillip A.

TK9.C575 2005
621.3'03--dc22

2004058572

T&F informa

Taylor & Francis Group
is the Academic Division of T&F Informa plc.

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

logic style. In the latter case, the size is only half of the static version, but the information is lost (due to leakage) after some time, needing "refresh" (or storing new data) for proper operation.

float switch a switch that is operated by a fluid level in a tank or process channel.

floating point characteristic part of a number that represents the exponent.

floating point operation arithmetic operation (e.g., add or multiply) involving floating point numbers (i.e., numbers with decimal points).

floating-point operations per second (FLOP) a measure of processing speed, as in megaFLOPs or gigaFLOPs.

floating-point register a register that holds a value that is interpreted as being in floating-point format.

floating-point representation in floating-point notation, a number is represented as a fractional part times a selected base (radix) raised to a power. This is the counterpart of scientific notation used in digital systems. The decimal equivalent of the floating-point value can be written $N.n = f \times r^e$ where f is the fraction or mantissa and e is a positive or negative integer called the exponent. *Contrast with* fixed-point representation.

floating-point unit a circuit that performs floating point computations, which is generally addition, subtraction, multiplication, or division.

FLOP *See* floating-point operations per second.

floppy disk a flexible plastic disk coated with magnetic material. Enclosed in a cardboard jacket having an opening where the read/write head comes into contact with the diskette. A hole in the center of the floppy allows a spindle mechanism in the disk drive to position and rotate the diskette. A floppy disk is a smaller, simpler, and cheaper form of disk storage media, and is also easily removed for transportation.

Floquet mode a solution to Maxwell's equations that can be supported by an infinite, periodic structure.

Floquet's theorem a basic theorem underlying the theory of wave propagation in periodic structures.

flow dependency *See* true data dependency.

flow diagram *See* flowchart.

flowchart a traditional graphic representation of an algorithm or a program, in using named functional blocks (rectangles), decision evaluators (diamonds), and I/O symbols (paper, disk) interconnected by directional arrows which indicate the flow of processing. Also called flow diagram.

flower pot a cover for the bushing of a pad-mount transformer.

fluidized bed combustion a method of solid-fuel combustion in which the fuel, usually coal, is pulverized and mixed with a ballasting substance and burned on a bed of pressurized air. If the ballasting agent is crushed limestone, sulfur from the coal is absorbed and carried out as solid ash.

fluorescence emission of light from an electronically excited state that was produced by absorption of radiation with a wavelength shorter than the emitted light. Fluorescence emission is a quantum mechanically allowed transition between electronic levels of the same spin state, resulting in emission of light with a very short lifetime, typically nanoseconds.

fluorescent lamp typically a lamp made by exciting a low pressure discharge in mercury vapor and other gases; mercury, when excited in the

Microsoft

OVER 10,000 ENTRIES

Microsoft®

Computer Dictionary

Fifth Edition

- *Fully updated with the latest technologies, terms, and acronyms*
- *Easy to read, expertly illustrated*
- *Definitive coverage of hardware, software, the Internet, and more!*

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Computer Dictionary.--5th ed.
p. cm.
ISBN 0-7356-1495-4
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002
004'.03--dc21 200219714

Printed and bound in the United States of America.

2 3 4 5 6 7 8 9 QWT 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com.*

Acquisitions Editor: Alex Blanton
Project Editor: Sandra Haynes

Body Part No. X08-41929

Usually a combination of hardware and software, a firewall prevents computers in the organization's network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization's network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization's network. *See also* proxy server.

firewall sandwich *n.* The use of load-balancing appliances on both sides of Internetworked firewalls to distribute both inbound and outbound traffic among the firewalls. The firewall sandwich architecture helps to prevent firewalls from degrading network performance and creating a single point of network failure. *See also* firewall, load balancing.

FireWire *n.* A high-speed serial bus from Apple that implements the IEEE 1394 standard. *See also* IEEE 1394.

firmware *n.* Software routines stored in read-only memory (ROM). Unlike random access memory (RAM), read-only memory stays intact even in the absence of electrical power. Startup routines and low-level input/output instructions are stored in firmware. It falls between software and hardware in terms of ease of modification. *See also* RAM, ROM.

FIR port *n.* Short for fast infrared port. A wireless I/O port, most common on a portable computer, that exchanges data with an external device using infrared light. *See also* infrared, input/output port.

FIRST *n.* Acronym for Forum of Incident Response and Security Teams. An organization within the Internet Society (ISOC) that coordinates with CERT in order to encourage information sharing and a unified response to security threats. *See also* CERT, Internet Society.

first-generation computer *n. See* computer.

first in, first out *n.* A method of processing a queue, in which items are removed in the same order in which they were added—the first in is the first out. Such an order is typical of a list of documents waiting to be printed. *Acronym:* FIFO. *See also* queue. *Compare* last in, first out.

first normal *n. See* normal form (definition 1).

fishbowl *n.* A secure area within a computer system in which intruders can be contained and monitored. A fishbowl is typically set up by a security administrator to impersonate important applications or information so that the system administrator can learn more about hackers who have broken into the network without the hacker learning more about or damaging the system. *See also* honeypot.

fitting *n.* The calculation of a curve or other line that most closely approximates a set of data points or measurements. *See also* regression analysis.

five-nines availability *n.* The availability of a system 99.999 percent of the time. *See* also high availability.

FIX *n.* Acronym for Federal Internet Exchange. A connection point between the U.S. government's various internets and the Internet. There are two Federal Internet Exchanges: FIX West, in Mountain View, California; and FIX East, in College Park, Maryland. Together, they link the backbones of MILNET, ESnet (the TCP/IP network of the Department of Energy), and NSInet (NASA Sciences Internet) with NSFnet. *See also* backbone (definition 1), MILNET, NSFnet, TCP/IP.

fixed disk *n. See* hard disk.

fixed-length field *n.* In a record or in data storage, a field whose size in bytes is predetermined and constant. A fixed-length field always takes up the same amount of space on a disk, even when the amount of data stored in the field is small. *Compare* variable-length field.

fixed-pitch spacing *n. See* monospacing.

fixed-point arithmetic *n.* Arithmetic performed on fixed-point numbers. *See also* fixed-point notation.

fixed-point notation *n.* A numeric format in which the decimal point has a specified position. Fixed-point numbers are a compromise between integral formats, which are compact and efficient, and floating-point numeric formats, which have a great range of values. Like floating-point numbers, fixed-point numbers can have a fractional part, but operations on fixed-point numbers usually take less time than floating-point operations. *See also* floating-point notation, integer.

fixed space *n.* A set amount of horizontal space used to separate characters in text—often, the width of a numeral in a given font. *See also* em space, en space, thin space.

fixed spacing *n. See* monospacing.

fixed storage *n.* Any nonremovable storage, such as a large disk that is sealed permanently in its drive.

fixed-width font *n. See* monospace font.

fixed-width spacing *n. See* monospacing.

functionality into limited space. Unlike most other programming languages, Forth uses postfix notation for its mathematical expressions and requires the programmer to work with the program stack directly. *See also* 4GL, interpreted language, postfix notation, stack, threading.

FORTRAN or **Fortran** *n.* Short for **formula translation**. The first high-level computer language (developed over the period 1954–58 by John Backus) and the progenitor of many key high-level concepts, such as variables, expressions, statements, iterative and conditional statements, separately compiled subroutines, and formatted input/output. FORTRAN is a compiled, structured language. The name indicates its roots in science and engineering, where it is still used heavily, although the language itself has been expanded and improved vastly over the last 35 years to become a language that is useful in any field. *See also* compiled language, structured programming.

fortune cookie *n.* A proverb, prediction, joke, or other phrase chosen at random from a collection of such items and output to the screen by a program. Fortune cookies are sometimes displayed at logon and logoff times by UNIX systems.

forum *n.* A medium provided by an online service or BBS for users to carry on written discussions of a particular topic by posting messages and replying to them. On the Internet, the most widespread forums are the newsgroups in Usenet.

Forum of Incident Response and Security Teams *n. See* FIRST.

forward *vb.* In e-mail, to send a received message, either modified or in its entirety, to a new recipient.

forward chaining *n.* In expert systems, a form of problem solving that starts with a set of rules and a database of facts and works to a conclusion based on facts that match all the premises set forth in the rules. *See also* expert system. *Compare* backward chaining.

forward error correction *n.* In communications, a means of controlling errors by inserting extra (redundant) bits into a stream of data transmitted to another device. The redundant bits are used by the receiving device in detecting and, where possible, correcting errors in the data. *See also* error-correction coding.

forward pointer *n.* A pointer in a linked list that contains the address (location) of the next element in the list.

FOSDIC *n.* Acronym for **f**ilm **o**ptical **s**ensing **d**evice for **i**nput to **c**omputers. A device used by the U.S. government to read documents on microfilm and store them digitally on magnetic tape or on a disk that can be accessed by a computer.

Fourier transform *n.* A mathematical method, developed by the French mathematician Jean-Baptiste-Joseph Fourier (1768–1830), for signal processing and signal generation tasks such as spectral analysis and image processing. The Fourier transform converts a signal value that is a function of time, space, or both into a function of frequency. The inverse Fourier transform converts a function of frequencies into a function of time, space, or both. *See also* fast Fourier transform.

four-nines availability *n.* The availability of a system 99.99 percent of the time. *See* high availability.

fourth-generation computer *n. See* computer.

fourth-generation language *n. See* 4GL.

fourth normal form *n. See* normal form (definition 1).

FPD *n. See* full-page display.

FPGA *n.* Acronym for **F**ield **P**rogrammable **G**ate **A**rray. A type of programmable logic chip that can be configured for a wide range of specialized applications after manufacture and delivery. FPGAs can be reprogrammed to incorporate innovations and upgrades. Because of their flexibility and adaptability, FPGAs are used in devices from microwave ovens to supercomputers.

FPLA *n. See* field-programmable logic array.

FPM RAM *n. See* page mode RAM.

FPU *n.* Acronym for **f**loating-**p**oint **u**nit. A circuit that performs floating-point calculations. *See also* circuit, floating-point operation.

FQ *n. See* fair queuing.

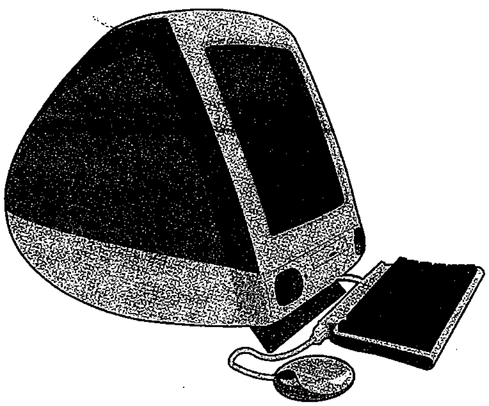
fractal *n.* A word coined by mathematician Benoit Mandelbrot in 1975 to describe a class of shapes characterized by irregularity, but in a way that evokes a pattern. Computer graphics technicians often use fractals to generate naturelike images such as landscapes, clouds, and forests. The distinguishing characteristic of fractals is that they are "self-similar"; any piece of a fractal, when magnified, has the same character as the whole. The standard analogy is that of a coastline, which has a similar structure whether viewed on a local or continental scale. Interestingly, it is often difficult to measure the length of the perimeter of such a shape exactly because the total distance measured depends on the size of the smallest element measured. For example, one could measure on a given coastline the

illegal operation might be impossible for a program or system because of built-in constraints. *Compare* invalid.

illuminance *n.* **1.** The amount of light falling on, or illuminating, a surface area. **2.** A measure of illumination (such as watts per square meter) used in reference to devices such as televisions and computer displays. *Compare* luminance.

IM *n. See* instant messaging.

iMac *n.* A family of Apple Macintosh computers introduced in 1998. Designed for nontechnical users, the iMac has a case that contains both the CPU and the monitor and is available in several bright colors. The "i" in iMac stands for Internet; the iMac was designed to make setting up an Internet connection extremely simple. The first version of the iMac included a 266-MHz PowerPC processor, a 66-MHz system bus, a hard drive, a CD-ROM drive, and a 15-inch monitor, with a translucent blue case. Later iMacs came with faster processors and a choice of case colors. See the illustration. *See also* Macintosh.



iMac.

.image *n.* A file extension for a Macintosh Disk Image, a storage type often used on Apple's FTP software download sites.

image *n.* **1.** A stored description of a graphic picture, either as a set of brightness and color values of pixels or as a set of instructions for reproducing the picture. *See also* bit map, pixel map. **2.** A duplicate, copy, or representation of all or part of a hard or floppy disk, a section of memory or hard drive, a file, a program, or data. For example, a RAM disk can hold an image of all or part of a disk in main memory; a virtual RAM program can create an image of some portion of the computer's main memory on disk. *See also* RAM disk.

image-based rendering *n. See* immersive imaging.

image color matching *n.* The process of image output correction to match the same colors that were scanned or input.

image compression *n.* The use of a data compression technique on a graphical image. Uncompressed graphics files tend to use up large amounts of storage, so image compression is useful to conserve space. *See also* compressed file, data compression, video compression.

image compression dialog component *n.* An application programming interface that sets parameters for compressing images and image sequences in QuickTime, a technology from Apple for creating, editing, publishing, and viewing multimedia content. The component displays a dialog box as a user interface, validates and stores the settings selected in the dialog box, and oversees the compression of the image or images based on the selected criteria.

Image Compression Manager *n.* A major software component used in QuickTime, a technology from Apple for creating, editing, publishing, and viewing multimedia content. The Image Compression Manager is an interface that provides image-compression and image-decompression services to applications and other managers. Because the Image Compression Manager is independent of specific compression algorithms and drivers, it can present a common application interface for software-based compressors and hardware-based compressors and offer compression options so that it or its application can use the appropriate tool for a particular situation. *See also* QuickTime.

image compressor component *n.* A software component used by the Image Compression Manager to compress image data in QuickTime, a technology from Apple for creating, editing, publishing, and viewing multimedia content. *See also* Image Compression Manager, QuickTime.

image decompressor component *n.* A software component used by the Image Compression Manager to decompress image data in QuickTime, a technology from Apple for creating, editing, publishing, and viewing multimedia content. *See also* Image Compression Manager, QuickTime.

image editing *n.* The process of changing or modifying a bitmapped image, usually with an image editor.

achieve personalization and privacy concomitantly, OPS is based on the concept of a Personal Profile, which is stored on the individual's computer and contains the user's unique identification, demographic and contact data, and possibly content preferences. This information remains under the user's control and can be released wholly or in part to the requesting site. *Acronym:* OPS. *See also* cookie, digital certificate.

open shop *n.* A computer facility that is open to users and not restricted to programmers or other personnel. An open shop is one in which people can work on or attempt to solve computer problems on their own rather than handing them over to a specialist.

Open Shortest Path First *n. See* OSPF.

Open Software Foundation *n. See* OSF.

open source *n.* The practice of making the source code (program instructions) for a software product freely available, at no cost, to interested users and developers, even though they were not involved in creating the original product. The distributors of open source software expect and encourage users and outside programmers to examine the code in order to identify problems, and to modify the code with suggested improvements and enhancements. Widely used open source products include the Linux operating system and the Apache Web server.

open standard *n.* A publicly available set of specifications describing the characteristics of a hardware device or software program. Open standards are published to encourage interoperability and thereby help popularize new technologies. *See also* standard (definition 2).

open system *n.* **1.** In communications, a computer network designed to incorporate all devices—regardless of the manufacturer or model—that can use the same communications facilities and protocols. **2.** In reference to computer hardware or software, a system that can accept add-ons produced by third-party suppliers. *See also* open architecture (definition 1).

Open Systems Interconnection reference model *n. See* ISO/OSI reference model.

OpenType *n.* A collaborative initiative by Microsoft and Adobe to unify support for Microsoft TrueType and Adobe PostScript Type 1 fonts. The OpenType font format enables font creators and users to work with the font type that best suits their needs without having to worry about whether the font is based on TrueType or PostScript technology. *Also called:* TrueType Open version 2. *See also* PostScript font, TrueType.

Opera *n.* A Web browser developed by Opera Software S/A. Opera is notable for its strict W3C standards support. Opera is often chosen by Web developers to test Web sites for W3C compliance. *See also* W3C, Web browser.

operand *n.* The object of a mathematical operation or a computer instruction.

operating system *n.* The software that controls the allocation and usage of hardware resources such as memory, central processing unit (CPU) time, disk space, and peripheral devices. The operating system is the foundation software on which applications depend. Popular operating systems include Windows 98, Windows NT, Mac OS, and UNIX. *Acronym:* OS. *Also called:* executive.

operation *n.* **1.** A specific action carried out by a computer in the process of executing a program. **2.** In mathematics, an action performed on a set of entities that produces a new entity. Examples of mathematical operations are addition and subtraction.

operation code *n.* The portion of a machine language or assembly language instruction that specifies the type of instruction and the structure of the data on which it operates. *Also called:* opcode. *See also* assembly language, machine code.

operations research *n.* The use of mathematical and scientific approaches to analyze and improve efficiency in business, management, government, and other areas. Developed around the beginning of World War II, operations research was initially used to improve military operations during the war. The practice later spread to business and industry as a means of breaking down systems and procedures and studying their parts and interactions to improve overall performance. Operations research involves use of the critical path method, statistics, probability, and information theory.

operator *n.* **1.** In mathematics and in programming and computer applications, a symbol or other character indicating an operation that acts on one or more elements. *See also* binary[1], unary. **2.** A person who controls a machine or system such as a computer or telephone switchboard.

operator associativity *n.* A characteristic of operators that determines the order of evaluation in an expression

rlogin[1] *n.* **1.** A protocol used to log in to a networked computer in which the local system automatically supplies the user's login name. *See also* communications protocol, logon. *Compare* telnet1. **2.** A UNIX command in BSD UNIX that enables a user to log in to a remote computer on a network using the rlogin protocol. *See also* BSD UNIX.

rlogin[2] *vb.* To connect to a networked computer using the rlogin protocol.

RLSD *n.* Acronym for **R**eceived **L**ine **S**ignal **D**etect. *See* DCD.

RMI-IIOP *n.* Acronym for **R**emote **M**ethod **I**nvocation over **I**nternet **I**nter-**O**RB **P**rotocol. A subsystem of the Java 2 Platform, Enterprise Edition (J2EE). It provides the ability to write CORBA applications for the Java platform without learning the CORBA Interface Definition Language (IDL). RMI-IIOP includes the full functionality of a CORBA Object Request Broker and allows the programming of CORBA servers and applications via the RMI application programming interface (API). RMI-IIOP is useful for developers using Enterprise Java Beans (EJBs), since the remote object model for an EJB is RMI-based. *Also called:* RMI over IIOP. *See also* CORBA, Enterprise JavaBeans, J2EE.

RMM *n. See* real-mode mapper.

RMON *n.* Acronym for **r**emote **mon**itoring or **r**emote network **mon**itoring. A protocol that enables network information to be monitored and analyzed at a central site. The nine management information bases (MIBs) defined by RMON provide statistics about network traffic. *See also* MIB. *Compare* SNMP.

roaming user profile *n.* A server-based user profile that is downloaded to the local computer when a user logs on; it is updated both locally and on the server when the user logs off. A roaming user profile is available from the server when logging on to a workstation or server computer. When logging on, the user can use the local user profile if it is more current than the copy on the server. *See also* local user profile, mandatory user profile, user profile.

robopost *vb.* To post articles to newsgroups automatically, usually by means of a bot. *See also* bot (definition 3), newsgroup, post.

robot *n.* **1.** A machine that can sense and react to input and cause changes in its surroundings with some degree of intelligence, ideally without human supervision. Although robots are often designed to mimic human movements in carrying out their work, they are seldom humanlike in appearance. Robots are commonly used in manufacturing products such as automobiles and computers. *See also* robotics. **2.** *See* bot, spider.

robotics *n.* The branch of engineering devoted to the creation and training of robots. Roboticists work within a wide range of fields, such as mechanical and electronic engineering, cybernetics, bionics, and artificial intelligence, all toward the end of endowing their creations with as much sensory awareness, physical dexterity, independence, and flexibility as possible. *See also* artificial intelligence, bionics, cybernetics.

robust *adj.* Able to function or to continue functioning well in unexpected situations.

ROFL *n.* Acronym for **r**olling **o**n the **f**loor, **l**aughing. An expression, used mostly in newsgroups and online conferences, to indicate one's appreciation of a joke or other humorous circumstance. *Also called:* ROTFL.

role-playing game *n.* A game that is played on line, such as MUD, in which participants take on the identities of characters who interact with each other. These games often have a fantasy or science fiction setting and a set of rules that all players need to follow. Role-playing games may be similar to adventure games in terms of story line, but also feature management and decision making for the character assumed during the course of the game. *Acronym:* RPG. *See also* MUD. *Compare* adventure game.

rollback *n.* **1.** A return to a previous stable condition, as when the contents of a hard disk are restored from a backup after a destructive hard disk error. **2.** The point in an online transaction when all updates to any databases involved in the transaction are reversed.

rollover *n. See* Year 2000 rollover.

ROM *n.* **1.** Acronym for **r**ead-**o**nly **m**emory. A semiconductor circuit into which code or data is permanently installed by the manufacturing process. The use of this technology is economically viable only if the chips are produced in large quantities; experimental designs or small volumes are best handled using PROM or EPROM. **2.** Acronym for **r**ead-**o**nly **m**emory. Any semiconductor circuit serving as a memory that contains instructions or data that can be read but not modified (whether placed there by manufacturing or by a programming process, as in PROM and EPROM). *See also* EEPROM, EPROM, PROM.

continuing through referrals to name servers at lower levels of the hierarchy, the DNS is able to match a "friendly" Internet address, such as microsoft.com, with its numerical counterpart, the IP address. Root servers thus contain the data needed for referrals to name servers at the highest level of the hierarchy. There are 13 root servers in the world, located in the United States, the United Kingdom, Sweden, and Japan. *Also called:* root name server. *See also* DNS (definition 1), DNS server, top-level domain.

root web *n.* The default, top-level web provided by a Web server. To access the root web, you supply the URL of the server without specifying a page name or subweb.

ROT13 encryption *n.* A simple encryption method in which each letter is replaced with the letter of the alphabet 13 letters after the original letter, so that A is replaced by N, and so forth; N, in turn, is replaced by A, and Z is replaced by M. ROT13 encryption is not used to protect messages against unauthorized readers; rather, it is used in newsgroups to encode messages that a user may not want to read, such as sexual jokes or spoilers. Some newsreaders can automatically perform ROT13 encryption and decryption at the touch of a key.

rotary dialing *n.* The signaling system used in telephones with rotary dials, in which each digit is associated with a set number of pulses. During dialing, these pulses, which are audible as series of clicks, momentarily turn the current in the telephone wires on and off. *Also called:* pulse dialing. *Compare* touch tone dialing.

rotate *vb.* **1.** To turn a model or other graphical image so that it is viewed at a different angle. **2.** To move bits in a register to the left or to the right. The bit that moves out of the end position rotates to the newly vacated position at the opposite end of the register. *Compare* shift.

R

rotational delay *n.* The time required for a desired disk sector to rotate to the read/write head. *Also called:* rotational latency.

rotational latency *n. See* rotational delay.

RO terminal *n.* Short for read-only **terminal**. A terminal that can receive data but cannot send data. Nearly all printers can be classified as RO terminals.

ROTFL *n. See* ROFL.

round *vb.* To shorten the fractional part of a number, increasing the last remaining (rightmost) digit or not, according to whether the deleted portion was over or under five. For example, 0.3333 rounded to two decimal places is 0.33, and 0.6666 is 0.67. Computer programs often round numbers, sometimes causing confusion when the resulting values do not add up "correctly." Percentages in a spreadsheet can thus total 99 percent or 101 percent because of rounding.

round robin *n.* A sequential, cyclical allocation of resources to more than one process or device.

roundtripping *n.* The process of converting files from one format to another for viewing or editing and then converting the files back to the original format again. In some cases, roundtripping can involve repeated conversions of the file from one format to another and back. Frequent roundtripping may be a concern because each conversion has the potential to introduce unwanted changes to the file.

routable protocol *n.* A communications protocol that is used to route data from one network to another by means of a network address and a device address. TCP/IP is an example of a routable protocol.

router *n.* An intermediary device on a communications network that expedites message delivery. On a single network linking many computers through a mesh of possible connections, a router receives transmitted messages and forwards them to their correct destinations over the most efficient available route. On an interconnected set of LANs (local area networks)—including those based on differing architectures and protocols—using the same communications protocols, a router serves the somewhat different function of acting as a link between LANs, enabling messages to be sent from one to another. *See also* bridge, gateway.

routine *n.* Any section of code that can be invoked (executed) within a program. A routine usually has a name (identifier) associated with it and is executed by referencing that name. Related terms (which may or may not be exact synonyms, depending on the context) are *function*, *procedure*, and *subroutine*. *See also* function (definition 3), procedure, subroutine.

routing *n.* The process of forwarding packets between networks from source to destination. *See also* dynamic routing, static routing.

Routing Information Protocol *n. See* RIP (definition 1).

routing table *n.* In data communications, a table of information that provides network hardware (bridges and routers) with the directions needed to forward packets of data to locations on other networks. The information contained

IBM

IBM Terminology

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #

Please send any feedback about the terms and definitions on this site to terms@ca.ibm.com

T

T1
: A digital trunking facility standard used in the United States and elsewhere, capable of transmitting and receiving 24 digitized voice or data channels. Signaling can be imbedded in the voice channel transmission when robbed-bit signaling is used. The transmission rate is 1544 kilobits per second. See also E1.

T1/D3
: A framing format used in T1 transmission.

T1/D4
: A framing format used in T1 transmission.

tabbable element
: In Web page creation, a page element that can be reached using the tab key.

tab character
: A character that indicates that printing or displaying should start at the next horizontal position on the current line. The tab is designated by '\t' in the C language and is named in the portable character set.

tab index
: In Web page creation, an attribute that allows the directed use of tab stops to change the default navigation through a page.

table
: (1) In a relational database, a database object that consists of a specific number of columns and is used to store an unordered set of rows. See also base table, temporary table, view.
(2) An orderly arrangement of data in rows and columns that can contain numbers, text, or a combination of both.
(3) In COBOL, a set of logically consecutive data items that are defined in the Data Division with the OCCURS clause.
(4) In RPG, a series of elements with like characteristics. A table can be searched for a uniquely identified element, but elements in a table cannot be accessed by their position relative to other elements.

table analysis
: An analysis process that consists of primary key analysis and the assessment of multicolumn primary keys and potential duplicate values.

table builder services message (TBSM)
: A message issued by a table builder module.

table check constraint
: See check constraint.

table collocation
: In a partitioned database environment, a state that occurs when two tables that have the same number of compatible partitioning keys are stored in the same database partition group. When this happens, the DB2 database management system can perform the join or subquery processing at the database partition where the data is stored.

uniqueness threshold
: A column analysis setting that infers whether a column is unique.

unique product
: A product that is uniquely identified to the i5/OS operating system by a product identifier (product ID) and version, release, and modification identifiers (Vx, Rx, Mx).

unique-weight sort sequence
: A sort sequence in which each graphic character in the sequence has a weight different from the weight of every other graphic character in the sequence.

unit
: (1) The defined space within disk units that is addressed by the system.
: (2) A mechanical, electrical, or electronic piece of equipment for a special purpose.

unit address
: (1) The address of a particular device, specified at the time a system is installed.
: (2) In mainframe computing, the address associated with a device on a given control unit. On ESCON or FICON interfaces, the unit address is the same as the device address. On OEMI interfaces, the unit address specifies a control unit and device pair on the interface.
: (3) The identifier for a logical subsystem and the logical device within the subsystem.

unit control block (UCB)
: A control block in common storage that describes the characteristics of a particular I/O device on the operating system. See also actual UCB, captured UCB.

United Nations Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT)
: An international set of electronic data interchange (EDI) standards published by the United Nations that is built upon X12 and TDI (Trade Data Interchange) standards.

United Nations Standard Products and Services Classification (UNSPSC)
: An open global standard for classifying products and services based on common function, purpose, and task.

United Nations Trade Data Interchange (UNTDI)
: A standard that preceded the UN/EDIFACT EDI standard.

unit name
: See device name.

unit number
: The unique identifier of a storage unit within a disk unit or a disk unit subsystem configured on the system.

unit of compilation
: In VS COBOL II, a section of source input from which the compiler produces a single object program. A unit of compilation can consist of a containing program and other programs nested within it.

unit of recovery (UR)
: (1) A recoverable sequence of operations within a single resource manager, such as an instance of DB2 for z/OS. See also unit of work, transaction.
: (2) A sequence of operations within a unit of work between sync points.
: (3) A defined package of work to be performed by the Resource Recovery Services (RRS).

unit of recovery descriptor (URD)
: A CICS control block that describes the progress of a unit of work through the sequence of syncpoint messages. The URD is chained off the CSA, and survives any failure of either system. It is used for recovery at CICS restart.

unit of recovery identifier (URID)

CMOS floating-point unit for the S/390 Parallel Enterprise Server G4

by E. M. Schwarz
L. Sigal
T. J. McPherson

The S/390® floating-point unit (FPU) on the fourth-generation (G4) CMOS microprocessor chip has been implemented in a CMOS technology with a 0.20-μm effective channel length and has been demonstrated at more than 400 MHz. The microprocessor chip is 17.35 by 17.30 mm in size, and one copy of the FPU including the dataflow and control flow but not including the FPR register file is 5.3 by 4.7 mm in size. There are two copies on the chip for error-detection purposes only; both copies execute the same instruction stream and are checked against each other. The high-performance implementation has a throughput of one instruction per cycle and an average latency of three execution cycles, yielding approximately 70 MFLOPS at 300 MHz on the Linpack benchmark. Currently, the G4 FPU is the highest-performance S/390 CMOS FPU with fault tolerance. It uses several innovative and high-performance algorithms not commonly found in S/390 FPUs or other FPUs, such as a radix-8 Booth multiplier, a Goldschmidt division and square-root algorithm, techniques for updating the exponent in parallel with normalization, and avoidance of the remainder comparison in quadratically converging division and square-root algorithms. Also demonstrated is a practical design technique for designing control flow into the dataflow and early floorplanning techniques.

Introduction

The IBM S/390* floating-point architecture is an extension of the well-known System/360* architecture from the 1960s [1]. The floating-point format has a hexadecimal exponent with a 7-bit characteristic biased by 64 and a 1-bit sign, as indicated by the following:

$$X = (-1)^{X_s} * 16^{(X_c-64)} * X_f, \qquad 0.0 \le X_f < 1.0,$$

where X_s is the sign bit, X_c is the characteristic, and X_f is the fraction or mantissa. Extended format was added in the 1970s along with square-root operation, which started out as a mathematical assist until it was recently included in the base architecture. The short format has a fraction of 24 bits, the long format has one of 56 bits, and the extended format has one of 112 bits. The G4 FPU is optimized for long format but also supports the other formats. The short format requires use of a trivial set of masking functions to implement on the long-format dataflow. The extended-format data are partitioned into two long-format numbers per operand, which requires several passes through the FPU to execute. The extended-format operations and high-order arithmetic operations

0018-8646/97/$5.00 © 1997 IBM

such as division and square root operate in a nonpipelined mode. They require many suboperations to complete and internally are highly pipelined but are not pipelined at the instruction level. The most common operations such as load, addition, and multiplication are pipelined and can execute one every cycle. The dataflow has been optimized for addition and multiplication of long-format operands, and very little hardware has been added to support nonpipelined instructions.

The G4 FPU is also responsible for performing fixed-point multiplication and fixed-point division. The FPU has a very fast multiplier which is capable of supporting two's-complement numbers for fixed-point calculations and sign-magnitude numbers for floating-point calculations. The fixed-point arithmetic operations are executed in a nonpipelined mode, which simplifies the data dependency analysis between instructions. The G4 microprocessor issues one instruction per cycle in order and completes at most one instruction per cycle in order. Thus, executing fixed-point multiply in nonpipelined mode does not cause much performance degradation, since most fixed-point instructions require only one execution cycle, and a fixed-point instruction stream would have to wait for the multiply result even if pipelined. This would result in a larger performance degradation in an out-of-order-completion machine. Fixed-point division is also executed in the FPU and uses an algorithm similar to floating-point divide short. However, there is the additional complexity in producing a remainder and conditionally complementing the input operands and output operands.

Also, the fixed-point unit (FXU) performs some functions normally thought to reside within the FPU. The FXU aligns input data from memory for floating-point instructions which are in RX format (register-and-indexed-storage operations). Data in memory are byte-addressable, and floating-point data can be 4 or 8 bytes, which are not necessarily aligned to a cache-line boundary. The cache returns the doublewords of memory containing the data and does not separate and rotate the data for the functional units. The operand buffers in the FXU provide this service for both the FXU and the FPU. The FXU also performs floating-point stores for the FPU, and that activity can require storage alignment, data masking, and multiple data writes that cross doubleword boundaries. The performance penalty of the FXU performing the floating-point stores is zero cycles for non-data-dependent stores, but two cycles for dependent stores.

The G4 FPU executes the most common floating-point instructions in a pipelined fashion with a throughput of one per cycle, and the infrequent operations are executed in a nonpipelined mode. The dataflow is described in detail, along with the execution of each type of instruction. In addition, the overall control flow is presented, followed by circuit implementation, physical design, discussion of designing control flow into the dataflow, and early floorplanning techniques.

Dataflow and execution

The fraction dataflow, shown in **Figure 1**, consists of a long-format multiplication and addition dataflow with a common 120-bit carry-propagate adder. At the top of the figure are the buses into and out of the FPU. The FPU_A_BUS and FPU_B_BUS are 64 bits each and bring operand 1 and operand 2 into the FPU from the FXU. Operand 1 is from the FPRs for floating-point computation and from the GPRs for fixed-point computation. Operand 2 is from the FPRs, the GPRs, or the operand buffers for memory operands. The operands are latched into the FPU A and B registers, which have fraction, exponent, and sign portions. Only the fraction part of the internal dataflow is shown in the figure. The output bus for the floating-point unit is the FPU_C_BUS, which is 64 bits wide and is driven to the register files. Internally there are eight additional bits of precision for intermediate calculations in the division and square-root routines. The FPU_C_BUS drives dependent data back into the A and B registers and into the first cycle of execution through the three late multiplexors. One other bus at the top of the dataflow is the output from the divide and square-root lookup tables. This bus is only 10 bits wide. All of these buses drive data to the A and B registers. The reading of operands from the register files or memory into these latches is defined to be the "E0" cycle. This is the cycle prior to the first execution cycle. Note that the A and B registers have multiplexors on their input which are capable of masking data for short-format instructions.

In the first execution cycle the A and B registers drive data into the late multiplexors. The term "late multiplexor" is actually a misnomer, since these multiplexors are located early in the first execution cycle, but can provide late-arriving interlocked data from the FPU_C_BUS. There is a late multiplexor for the multiply A operand (MAL), the adder A operand (AAL), and the B operand for both multiply and add (BL). The multiply A operand can be 64 bits to support extra precision for division and square-root intermediate calculations. The other late multiplexors are 56 bits to support long format.

The MAL and BL multiplexors drive the multiply first cycle, which consists of a 3× adder and Booth decode; for addition, the AAL and BL multiplexors drive the compare and swap and aligner and XOR logic. The MAL and BL multiplexors also drive binary shifters which are capable of binary-aligning data or forcing binary shifts of up to 3 bits right or left. These shifters are used for division and square-root operations, and their output is latched back into the A and B registers. The output of the multiply first cycle is latched in the 3× and × registers and the Booth

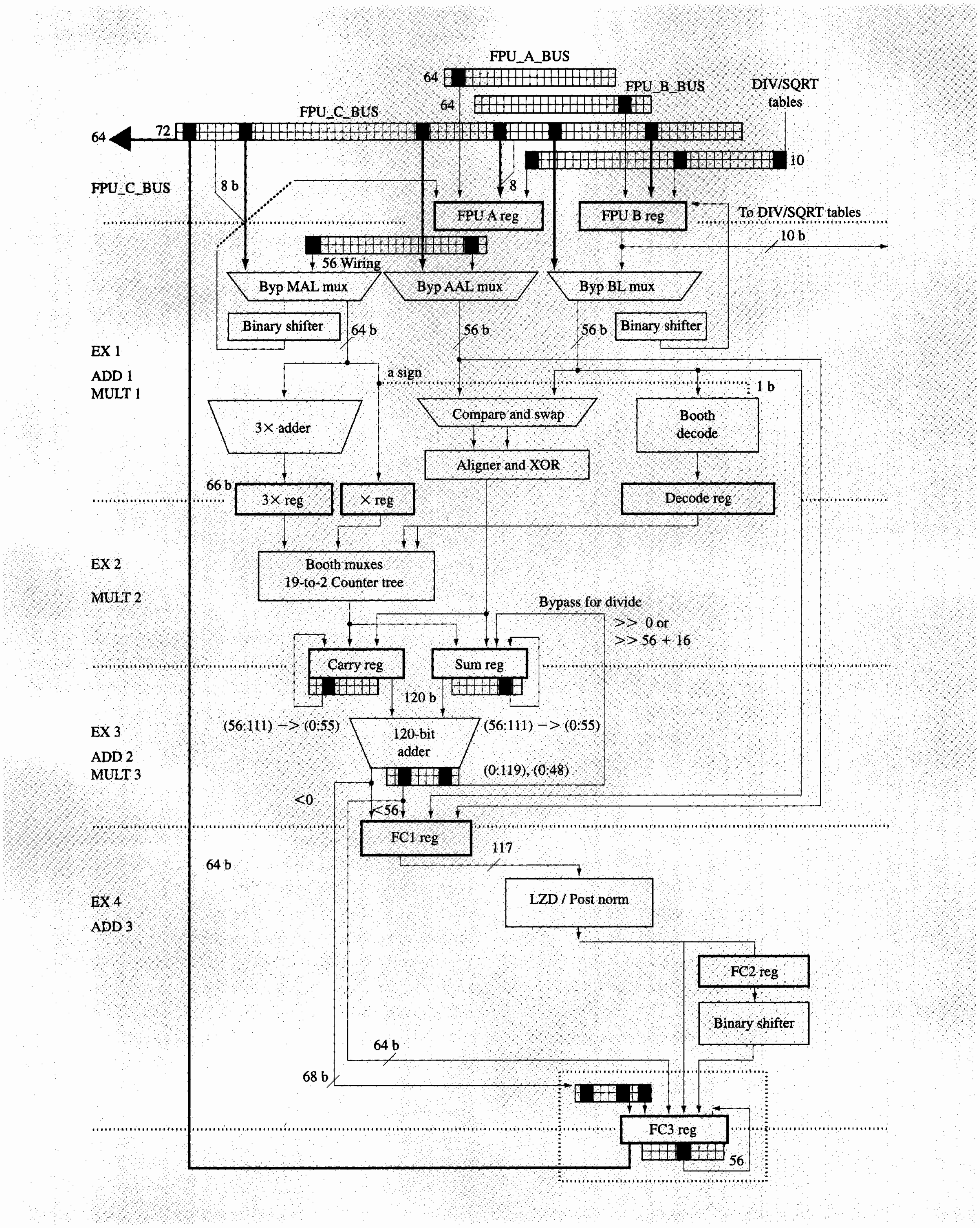


Figure 1

Fraction dataflow.

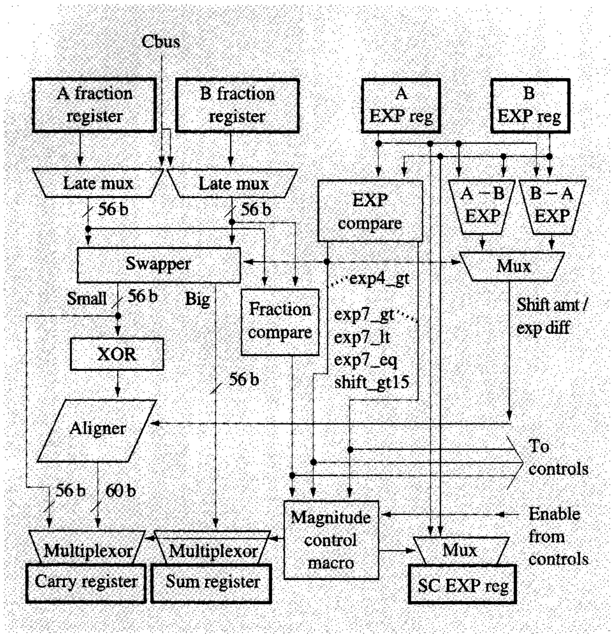


Figure 2

Dataflow of the first addition cycle.

decode registers. The output of the addition first cycle is latched in the carry and sum registers.

The output of the multiply's 3× register, × register, and Booth decode registers feeds a Booth multiplexor and a 19-to-2 counter tree resulting in two partial products. These two partial results are 120 bits each and are latched into the 120-bit carry and sum registers.

The second cycle of an addition and third cycle of a multiplication pass through the 120-bit adder. There are several feedback paths shown which are used for extended-precision operations and for division and square root. The result of the adder can be driven to either the FC1 register or the FC3 register (for the case of a multiply).

The FC1 register drives 117 bits to the post-normalizer for the third addition execution cycle. The post-normalizer determines the shift amount by performing a leading-zero detect (LZD) of the fraction, and then the fraction is shifted and the exponent is updated in parallel. The normalizer output is driven to both the FC2 and FC3 registers.

The FC2 register provides an extra internal working register for division and square root. It also drives to a binary shifter, which is used to transform binary-aligned intermediate results for division and square root into hex-aligned results. The binary shifter can shift the fraction up to 3 bits left or right and is controlled by an LZD of the most significant digit or by a forced shift amount from controls. The binary shifter output is connected to the FC3 register.

The FC3 register receives the result of the arithmetic computation and drives the results to register files or back to the FPU dataflow by way of the FPU_C_BUS. The FC3 register also has the ability to shift itself by 56 bits to the left so that extended data can be stored in the FC3 register and written to the FPU_C_BUS with two back-to-back write cycles without involving the other elements of the FPU fraction dataflow.

The overall FPU fraction dataflow has five stages, though the most common operations require only three cycles or stages of execution. The execution of addition, multiplication, load, division, square-root, and extended-precision instructions is detailed in the following subsections.

- *Addition*

The operations subtract, add, or compare are collectively referred to as an addition. Floating-point addition involves aligning the fractions of the operands, conditional complementation of the smaller operand, a two's-complement addition, normalization, and condition code setting. This usually requires three cycles of execution. The key to subtraction of sign-magnitude numbers is identifying and complementing the smaller of the two operands prior to the carry-propagate addition. The resulting sum is a magnitude and does not require conditional post-complementation. As shown in **Figure 2**, the first cycle consists of comparing the A and B register exponents to determine which operand is the smaller of the two, and conditionally swapping the fractions so that the smaller operand proceeds to be conditionally complemented and aligned by the exponent difference. One additional hex guard digit (4 bits) is maintained during alignment, as specified by ESA/390* floating-point architecture. Then the larger operand is placed in the sum register and the aligned operand is placed in the carry register. The second cycle involves a two's-complement addition of the fractions. The 28 fraction bits of short operands or 60 fraction bits of long operands are easily accommodated by the 120-bit adder. The sum is latched in the FC1 register. The third cycle of execution involves a post-normalization.

Post-normalization of the sum is not always necessary, but it is rather frequent for S/390 architecture. In all architectures there could be cancellation of the most significant bits for an effective subtract operation which changes the location of the most significant bit. There could also be a carry-out for an effective addition operation. But for S/390, there is the additional possibility of unnormalized fractions or the case of a zero fraction with a nonzero characteristic. Other architectures have

denormalized numbers, but the occurrence is less frequent and their range of values is limited. So, for simplicity all additions are routed through the post-normalizer.

In the post-normalization cycle, a leading-zero detect is performed on the fraction to determine the shift amount. The most significant bits of the shift amount are available earlier than the least significant bits, and the exponent is updated in parallel with the delayed arrival of the least significant bits [2]. The determination of exponent underflow and overflow is also calculated in parallel with the fraction normalization. The resulting normalized fraction and exponent are latched in the FC3 register at the end of the third cycle of execution. During the following cycle the result is written into the FPR.

Of these three execution cycles, the first addition cycle is the most complex; Figure 2 describes the interconnection of the dataflow. Since this cycle is very timing-critical, both the dataflow and the control design were specified with custom circuits. The cycle begins with the A and B exponent registers being driven to the exponent compare circuit and to the exponent difference logic. A signal called $EXP4\_GT$, which is a compare of A exponent greater than B for the least significant four bits of exponent, is driven to the swapper. It actually uses five bits of exponent for the comparison, as shown by the following equation:

$$EXP4\_GT = (EA_{4lsb} > EB_{4lsb}) \oplus (EA_5 \oplus EB_5).$$

If the fifth-to-least significant bits (EX_5) of the two exponents are not equal, the comparison of the least significant four bits ($EA_{4lsb} > EB_{4lsb}$) is inverted. Thus, $EXP4\_GT$ is an approximation of which operand is greater. The operand guessed to be larger is driven on the Big bus, and the smaller operand is driven on the Small bus. Feeding the swapper in the fraction dataflow are the late multiplexors, which can receive data from the FPU_C_BUS or the fraction registers. The Small bus drives an XOR circuit which conditionally complements the data for an effective subtract operation where the shift amount is less than 16. The XOR output drives an aligner which can shift right 0 to 15 digits. The shift amount is determined by two subtractors. Both exponents A minus B and B minus A are calculated, and then the appropriate result is chosen once $EXP4\_GT$ is known. The output of the aligner drives to the carry multiplexor/register. The Big signal is driven directly to the sum multiplexor/register. The selection of carry and sum multiplexors is determined by a custom control macro called the A1 magnitude control macro. It determines whether the swap was correct and whether to force zeros for shifts greater than the width of the fractions or to mask the operands for short data. In addition to the fraction being latched, the larger of the two exponents is latched into the sum and carry exponent registers (SC EXP reg).

This is the general procedure for most cases of addition execution, although the procedure is actually more complex and can be separated into five cases: effective load, effective add, one's-complement, two's-complement, and simple subtract. The details for each case are described below.

Effective load: ($SHIFT\_GT15$)
If there is an exponent difference greater than 15 (signaled by $SHIFT\_GT15$), the operation is effectively a load. There are 14 hex digits in a long operand, and S/390 dictates that one additional guard digit be maintained, for a total of 15 hex digits. Note that short operands were treated the same as longs, but with additional masking to allow only seven hex digits of precision. Since the rounding mode of S/390 hex floating-point is truncation with the exception of square root, the smaller operand does not contribute to the addition if the exponent difference is greater than 15.

The execution of this case of addition could be accomplished in two execution cycles. However, it was designed to be completed in three cycles to avoid creating any critical control paths. So, for simplicity this case consumes an adder cycle. To accomplish this, the operand with the larger exponent and zeros are gated into the sum and carry registers. There is a complication in doing this, since the swapper does not use an exact exponent greater than the compare signal but instead uses a signal based on the least significant four bits. Thus, there is the potential for the swap to be incorrect (i.e., the Big signal is actually the smaller of the two operands, and Small is the bigger operand). This must be taken into account in the A1 magnitude control macro when selecting the larger operand and zeros to be multiplexed into the two registers.

This case executes in three cycles, and in the first cycle, the operand with the greater exponent and zero is gated into the sum and carry registers.

Effective add: ($\overline{SHIFT\_GT15} \cdot \overline{EFF\_SUB}$)
Another simple case is an effective add operation (signaled by $\overline{EFF\_SUB}$) which exists when the operation is add and the operands have the same sign or the operation is subtract or compare and the operands have different signs. For this case, no complementation is necessary. Since the shift amount is less than 16, the least significant bits of the exponent are guaranteed to give the proper shift amount, and this 4-bit exponent difference is driven to the aligner. The aligner output and the Big bus are driven to the carry and sum registers, respectively. This case requires three cycles, since the normalization cycle may be required if the operand with the larger exponent is unnormalized.

Conditional one's-complement:
$(\overline{SHIFT\_GT15} \cdot EFF\_SUB \cdot EXP\_EQ)$
The remaining cases are more difficult and involve an effective subtraction with shift amount less than 16. Determination of the operand to conditionally complement and align is difficult, especially considering that the input fractions could be unnormalized. The case in which the exponents are equal (signaled by *EXP_EQ*) is called conditional one's-complement. For timing reasons, the fraction comparator receives the unaligned operands. When the exponents are equal, the fraction comparison gives a true indication of the relative magnitude of the two operands, but it is determined too late to complement the smaller of the two before addition. However, this is not necessary to obtain the correct result. In [3], a method is described of always complementing operand B and still computing the correct result. Note, for $A - B$,

$$|R| = |A| - |B| = |A| + \overline{|B|} + 1 \text{ ulp},$$

where $\overline{|B|}$ is the one's-complement of B and ulp refers to a unit in the last place of operand B. For $B - A$ the following can be derived [4]:

$$|R| = |B| - |A| = \overline{(|A| + \overline{|B|} + 0 \text{ ulp})}.$$

Thus, B can always be complemented, if for $B < A$ the carry-in to the adder is set (1 ulp) and the true sum is the result, and for $A < B$ the carry-in is zero and the complement of the sum is the result. This causes the critical path to be the setting of one bit, the carry-in to the adder, rather than requiring conditionally complementing all the fraction bits of Small and Big and conditionally selecting them to the carry and sum registers. The control signal for the FC1 register which receives the adder output can be determined to be the true or the complemented output.

Thus, this case requires three cycles of execution: the first cycle, in which B is complemented and the carry-in to the adder conditionally set based on the fraction greater than signal, the second cycle of a 2-to-1 addition and conditional selection of the true or complemented output into the FC1 register, and the third cycle of post-normalization.

Conditional two's-complement:
$(\overline{SHIFT\_GT15} \cdot EFF\_SUB \cdot \overline{EXP\_EQ} \cdot UNNORM)$
When the exponents are not equal and the data are unnormalized (signaled by *UNNORM*), this is called the conditional two's-complement. For this case, a prior determination of the larger operand is not possible, so a post-adder determination is made. Four cycles of execution are needed; two cycles use the adder. The second cycle through the adder is used to create a two's-complement of the sum. The first time through the adder, the carry-out can be latched along with the guessed true sum, and in the following cycle the carry-out can be used as a select signal either to hold the true sum if the original guess of the greater operand is correct, or to select the complemented sum to be gated into the FC1 register. Thus, there is a pipeline stall for this case. It was estimated that this case occurs only about 3 percent of the time.

Simple subtract:
$(\overline{SHIFT\_GT15} \cdot EFF\_SUB \cdot \overline{EXP\_EQ} \cdot \overline{UNNORM})$
If the input operands are both normalized and the shift is less than 16, the exponent compare of the least significant four bits truly indicates the smaller of the two operands. Thus, the correct operand can be identified and complemented in the first cycle of execution. This is called the simple subtract case, and requires only three cycles of execution.

Thus, in summary for addition, there exists only one rare case, conditional two's-complement, which requires four execution cycles; all of the other cases are completed in three execution cycles. The adder dataflow has been optimized to be pipelined one instruction per cycle, and each cycle has been optimized to meet cycle time. The most complex of these cycles is the first add cycle, which makes many of its decisions based on only four or five bits of the exponents. This enables a very fast cycle time, but at the cost of complexity of design, which is evident in the five separate cases of control signal selection. To reduce timing, the control design was implemented in custom circuits; it was designed into the dataflow early in the design phase.

- *Multiplication*

Both fixed-point and floating-point multiplication are performed on the FPU's multiplier. In addition, the multiplier is used by the division and square-root routines. These high-order routines require greater precision for intermediate results than is supported by the long format. Thus, additional bits of precision are necessary, but the critical timing of long format for a 56-by-56-bit multiplication must not be exceeded. To accomplish this, only one operand is extended with additional bits to be 64 bits. The other operand dictates the cycle time, since it specifies the number of partial products which determine the number of stages in the counter tree. Other designs [4] have increased both operands, creating a 60-by-58-bit multiplier which increases the performance of division and square root. On the G4 microprocessor chip, though, this type of implementation would create a longer cycle time, affecting the performance of all instructions. Thus, a 56-by-64-bit multiplier was implemented in the G4 FPU.

The counter tree design in the FPU is an important part of the FPU design; it is timing-critical and it is very large,

consuming 15 percent of the G4 FPU. Our design goal for the multiplier was to have it pipelined every cycle and have an approximate latency of two or three cycles. One full cycle is required for 2-to-1 addition, so the counter tree had to complete in one or two cycles. Radix-4 Booth algorithms [5, 6] are interesting because of their simplicity, but for a 56-bit operand they require a 29-to-2 partial product array, and Booth multiplexing requires a 2-to-1 true/complement multiplexor. This counter tree requires eight levels of 3/2 counters, which did not meet our cycle time objective. Another option is to implement the counter tree with latches in the middle, but this is prohibitive to implement because of the large number of signals that must be latched.

Two other alternatives are to create a two-cycle path that is unlatched or to perform an iterative multiply. The first option was rejected by our test personnel, who did not like the large number of paths that had to be specially tested for cycle time. This would have added a huge amount of time to test the chip, and was determined to be too costly. The other alternative is an iterative multiply [7] in which half the multiplication is computed each iteration, e.g. by using a 28-by-64-bit multiplier. This has a very small area and very fast critical path, but it does not have the performance benefit of being able to pipeline a multiply every cycle.

The solution that has been implemented in the G4 chip is a higher-radix Booth algorithm. In particular, a radix-8 Booth algorithm was used to simplify the counter tree [8, 9]. The counter tree for a radix-8 algorithm requires only a 19-to-2 partial product reduction which has six levels of 3/2 counters. The counters in 0.2-μm technology supporting a 300-MHz clock frequency have a delay of approximately 350 ps for the sum output and 250 ps for the carry output, for an average delay of 300 ps. The Booth multiplexing requires a 4-to-1 true/complement multiplexor which has a delay of approximately 450 ps. The delay of both the counter tree and the Booth multiplexing meets the cycle-time objective. The main problem with higher-radix algorithms, though, is creating the difficult multiples of the multiplicand which are not powers of 2. For a radix-8 algorithm, the multiples of $+4$, $+3$, $+2$, $+1$, 0, -1, -2, -3, and -4 of the multiplicand are required. The only difficult multiples to form are the $\pm 3\times$. In the first cycle of execution, the $3\times$ multiple is formed, and the Booth selects for the multiplexing are determined. In the second cycle, the Booth multiplexing between all of the possible multiples ($+4$ to -4) is performed using 4-to-1 true/complement multiplexors; then, 19 partial products are reduced to two in six levels of 3/2 counters. This cycle is shown in **Figure 3**. In the third cycle, the 2-to-1 addition is performed. A radix-8 Booth algorithm provides the best partitioning for our

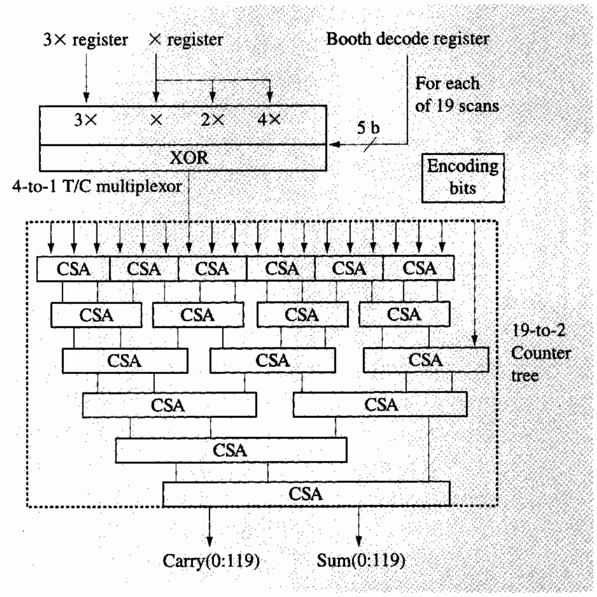


Figure 3

Dataflow of the second multiplication cycle.

particular implementation. The detailed implementation of the counter tree is described in [10].

The common cases for post-normalization have been designed into the dataflow of the third cycle of execution. From instruction traces, it was determined that more than 90 percent of the time both operands are normalized. If both operands are normalized, it can easily be shown that the result could have one of two normalizations. Multiplying the minimum normalized numbers and the maximum normalized numbers gives the following range of products:

$$(0.1)_{16} * (0.1)_{16} = (0.01)_{16}$$

and

$$(0.FFF\cdots)_{16} * (0.FFF\cdots)_{16} < (0.FFF\cdots)_{16}.$$

Thus, the minimum product requires a left shift of one hex digit (4 bits), and the maximum product has a shift of zero, or no shift. The determination of which shift is required is designed into the 120-bit adder to execute in parallel with the determination of the conditional sum of the most significant hex digit. An enable signal is sent to this logic to allow it to drive the select signals to the FC3 register. Both possible combinations of fraction shifts are driven to the FC3 register and are selected by this control signal, which was designed into the dataflow in custom circuits. Thus, the critical control paths were designed in custom logic. Note that if the operands are unnormalized

or if the exponents are close to underflowing or overflowing, the multiply instruction requires four cycles and is driven into the post-normalizer. Hence, the most common case has a latency of three cycles with a throughput of one instruction per cycle, but some cases have a latency of four cycles with a throughput of one per cycle.

- *Load*

Loads are executed in the floating-point pipeline to take advantage of the fast bypassing between data-dependent instructions. The actual execution involves two cycles. First the operand in the B register is moved directly to the FC1 register. The second cycle passes through the normalizer with a forced shift amount (no normalization is allowed by the architecture) and is latched into the FC3 register. Thus, loads require two execution cycles.

- *Division*

The division implementation uses the Goldschmidt algorithm, which has been described in many conferences and papers [4–6, 11–16]. It is a very interesting algorithm for high performance because of its quadratic convergence and its ability to execute some of its multiplications in parallel. The algorithm was first used in the IBM System/360* Model 91 [17], but since then has not been implemented on S/390 mainframes, except for one low-end mainframe [4]. The main limitation of this algorithm is that it requires nontrivial error analysis and the avoidance of extra cycles to round the result. Other algorithms such as nonrestoring algorithms or even the Newton–Raphson quadratically converging algorithm are much easier to analyze, since they are self-correcting. Analyzing the error in one iteration and the error in the lookup table is enough to prove these algorithms for the Nth iteration. The Goldschmidt algorithm has error which propagates each iteration, and all iterations must be analyzed.

The algorithm is based on converging the divisor (denominator) to one; then, the dividend (numerator) is equal to the quotient. Let Q equal the quotient, and the dividend of the ith iteration is N_i, the divisor D_i, and the convergence factor R_i. The following can be stated:

$$\begin{aligned} Q &= \frac{N_0}{D_0} \\ &= \frac{N_0}{D_0}\frac{R_0}{R_0} \\ &= \frac{N_0}{D_0}\frac{R_0}{R_0}\cdots\frac{R_{i-1}}{R_{i-1}} \qquad \text{if } D_i \approx 1.0, \\ &= \frac{N_i}{D_i}; \qquad \text{then } Q \approx \frac{N_i}{1.0} = N_i. \end{aligned}$$

The initial convergence factor is determined from a lookup table, but the subsequent convergence factors are determined by two's-complementing the current denominator:

$$\begin{aligned} R_0 &\approx \frac{1}{D_0} \\ D_1 &= D_0 * R_0 \\ &= D_0 * \left(\frac{1}{D_0} + E_{R0}\right) \\ &= 1 + D_0 E_{R0}; \end{aligned}$$

Let

$$\begin{aligned} X_1 &= -D_0 E_{R0}, \\ D_1 &= 1 - X_1, \\ R_1 &= 2 - D_1 = 2 - (1 - X_1) = 1 + X_1; \end{aligned}$$

Then,

$$\begin{aligned} D_i &= D_{i-1} * R_{i-1}, \\ N_i &= N_{i-1} * R_{i-1}, \\ R_i &= 2 - D_i. \end{aligned}$$

The error analysis was performed by 1) expanding the equations for four iterations, where RC_i is the calculated convergence factor in the ith iteration including error terms, DC_i is the calculated divisor, NC_i is the calculated dividend, $t_{\text{even}\,i}$ is the truncation error in the divisor calculation, $t_{\text{odd}\,i}$ is the truncation error in dividend calculation, E_{R0} is the error in the lookup table, and ts_i is the small error in the convergence factor truncation; and 2) performing a one's-complementation instead of a two's-complementation:

$$\begin{aligned} RC_0 &= 1/D + E_{R0}, \\ DC_0 &= D, \\ NC_0 &= N, \\ DC_i &= DC_{i-1} * RC_{i-1} - t_{2*i}, \\ NC_i &= NC_{i-1} * RC_{i-1} - t_{2*i-1}, \\ RC_i &= 2 - DC_i - ts_i. \end{aligned}$$

The analysis of the error in the final quotient, NC_4, was arduous. After substituting the values of the errors due to truncation, it was possible to prove that the implementation satisfied the error constraints.

The timing diagram of the iteration cycles of a Goldschmidt division algorithm is shown in **Table 1** for long operands. Short operands require only three iterations. In the G4 implementation there are an additional four cycles on the front end of the diagram to get the operands binary-normalized and in the proper

Table 1 Timing diagram for long divide.

Cycle	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
	R0 N0	D1	D1 N1	D1 N1	D2 N1	D2 N2	D2 N2	D3 N2	D3 N3	D3 N3	N3	PN4	PN4	PN4	N4

registers. And there are several cycles on the back end to hex-normalize and properly round the result.

Rounding poses an additional problem for quadratically converging algorithms, since a remainder is not determined as an intermediate product of the iteration step. The G4 implementation is able to eliminate the remainder comparison step half of the time by examining an additional bit of precision [17]. The intermediate result is formed with one additional guard bit with an error tolerance of less than the weight of the guard bit. For truncation which S/390 dictates, if the guard bit is 1, the result should be truncated. This is true since the error tolerance guarantees that the actual quotient is less than the next higher machine-representable number and not equal to it. But if the guard bit is 0, a remainder comparison is needed. If the remainder is greater than or equal to zero, the result is the truncated intermediate result; if the remainder is less than zero, the result is the decremented intermediate result. This eliminates the remainder comparison for half of the cases. If there are additional guard bits, this algorithm can be expanded to eliminate the remainder comparison in all but one of the 2^G cases, where G is the number of guard bits.

With the startup penalty of hex-normalizing and then binary-normalizing the operands and the ending penalty of hex-aligning the data, the overall division requires either 18 or 24 cycles for short operands and either 22 or 28 cycles for long operands for a guard bit equal to 1 or 0, respectively. The startup and ending penalties are very significant and make a quadratically converging algorithm only slightly better than a nonrestoring division algorithm. Thus, the G4 FPU chip implements a very aggressive division algorithm which is quadratically converging, and eliminates the remainder comparison in half of the cases.

- *Square root*

The square-root algorithm is also based on the Goldschmidt algorithm [18]. The following are the equations used, where r_i is the square root of the convergence factor in the ith iteration, SQr_i is the convergence factor, B_i is the accumulative approximation to the root of N, and X_i is an intermediate variable that approaches 1.

Initialize:

$$r_0 \approx 1/\sqrt{N},$$
$$B_0 = N,$$
$$X_0 = N;$$

iterate:

$$SQr_i = r_i * r_i,$$
$$B_{i+1} = B_i * r_i,$$
$$X_{i+1} = X_i * SQr_i,$$
$$r_{i+1} = 1 + 0.5 * \bar{X}^f_{i+1};$$

final:

$$\sqrt{N} \approx B_{\text{last}}.$$

The expression for the square root of the convergence factor, which does not take much delay to calculate, is one plus one half of the fractional part of X complemented. It is formed in the binary shifters located near the A and B fraction registers. The choice of the above formula for the square root of the convergence factor can be best understood by studying the convergence of X_{i+1} to 1.0. Let X_i be d_i different from 1.0; then,

$$\begin{aligned} X_i &= 1 - d_i, \\ r_i &\approx 1 + 1/2 * d_i \\ &\approx 1 + 1/2 * (1 - X_i) = 1 + 1/2 * (2 - X_i - 1) \\ &\approx 1 + 1/2 * (\bar{X}_i - 1) = 1 + 1/2 * \bar{X}^f_i, \\ SQ_i &\approx 1 + d_i + 1/4 * d_i^2 \approx 1 + d_i, \\ X_{i+1} &\approx 1 - d_i^2. \end{aligned}$$

Thus, there is quadratic convergence with this choice for the square root of the convergence factor.

The overall delay of the implementation is 26 or 32 cycles for short operands and 35 or 42 cycles for long operands, depending on the guard bit. This implementation eliminates the remainder calculation in half of the cases.

- *Extended-precision instructions*

Extended-precision instructions were also implemented in hardware, but in nonpipelined mode. Their performance is not critical, and thus uses simple cost-efficient algorithms.

Extended-precision addition is performed by examining the exponents and aligning the operand with the smaller

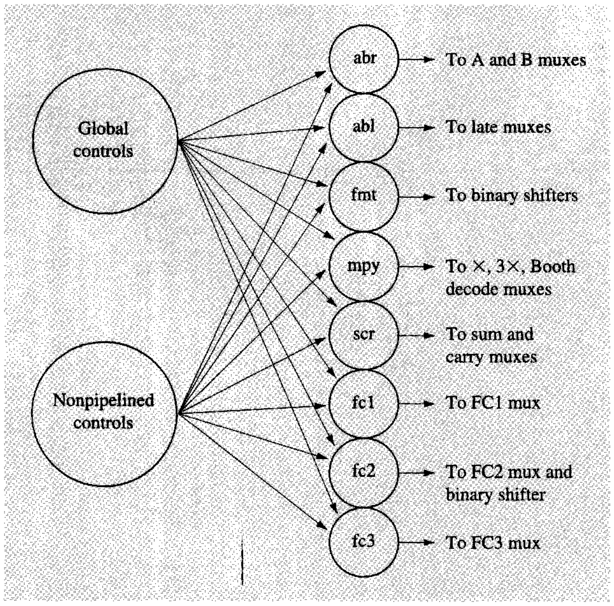


Figure 4

Overall control flow.

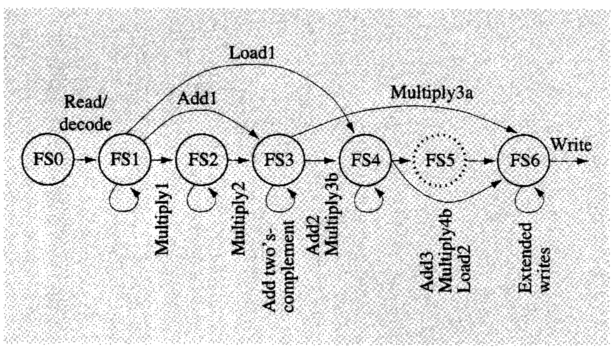


Figure 5

Pipeline state diagram.

exponent in the post-normalizer. The post-normalizer is 121 bits wide and has the capability of having the shift amount overridden by controls which then specify its own shift amount. Once data have been aligned, they are conditionally complemented and loaded in stages into the carry and sum registers. Then the actual addition requires one cycle. The result is then normalized and driven to the FC3 register, where it is written to the FPRs with two write cycles.

Extended-precision multiplication is performed by separating the extended inputs into two long fractions each. Four long multiplications and three additions are performed. Note that pre-normalization may be required if the operands are not already normalized.

Extended-precision division and square root use restoring 1-bit algorithms. Paths have been created into the carry and sum multiplexor/registers to support this. The latency is very long, but the amount of hardware invested is minimal.

Control flow

The control flow was designed with synthesized macros to allow changes late in the design phase for problems found in simulation. The overall control flow is shown in **Figure 4**. There are two major macros in the design: the global control macro and the nonpipelined control macro. The global control macro handles the state information for pipelined instructions and sequences the start of nonpipelined instructions. It also handles all handshaking with other units. The nonpipelined control macro handles the select lines for all nonpipelined instructions such as extended-precision floating-point, division, square-root, and fixed-point instructions. These routines are similar to horizontal millicode routines, since the macro implements a cycle counter and instruction decode which determine the selects to enable. The other control macros listed are collectively referred to as the pipeline-select macros; they determine the values on the select lines from global pipeline state information and from information from the nonpipelined control macro. This partitioning of controls makes it simpler to design, since the task of maintaining state information is separated from determining the select line values.

Global state information for pipelined instructions is maintained in the global control macro (see **Figure 5**). There are seven states: FS0, FS1, FS2, FS3, FS4, FS5, and FS6. FS0 corresponds to the E0 cycle, FS1 corresponds to the E1 cycle, FS2 corresponds to the E2 cycle of multiplication, FS3 corresponds to the 120-bit adder cycle, FS4 corresponds to the normalizer, FS5 is not maintained but corresponds to the shifter between the FC2 and FC3 registers, and FS6 corresponds to the write cycle. There are basically two methods for designing pipelines: up-front blocking and feed-forward. Up-front blocking prevents an instruction from entering the pipeline until it is guaranteed not to have any dependencies or resource conflicts. Feed-forward pushes the instruction as far into the pipeline as possible until contentions or dependencies make it wait. Since the G4 FPU can be considered a peripheral unit which is not aware of the global central processor state, it was best to implement a feed-forward pipeline so as not to become instruction-starved or data-starved.

In the global control macro there is a scoreboard-type implementation of the information for each state [19]. This information includes the write address, the length of the operands, whether the data must be normalized, the instruction type, etc. The status of each state is also maintained (e.g., whether the state is valid or whether it is busy and holding for this cycle). The state information from the global control macro is sent to pipeline-select macros, where decisions are made as to which selects to each multiplexor should be invoked. For example, if the FS3 state which corresponds to the 120-bit adder function is busy, the sum and carry registers should be held. This is determined by the sum and carry registers pipeline-select macro (SCR), which drives select lines to the multiplexor. Separating maintenance of the global state from determining values of select lines made the controls simpler to design.

Also, the global control macro is responsible for resolving resource conflicts. An example is a multiply instruction followed by an add instruction. Figure 5 shows that the multiply requires the FS1, FS2, and FS3 states, while the add requires the FS1, FS3, and FS4 states. There may be a contention for the FS3 state, which is the 120-bit addition cycle. In this case the add would be delayed, since the multiply was issued first. This conflict can be resolved by simply giving the transition from the FS2 to the FS3 state higher priority than the transition from the FS1 to the FS3 state. Note that this example shows a common resource conflict of using the adder for both multiplication and addition. The performance benefit of eliminating this conflict by having two adders was determined not to be worth the area cost.

- *Resolving data dependencies*

The most complex part of the control design is the resolution of data dependencies between instructions. These dependencies can be classified into four types, depending on timing and the buses used for bypassing (or wrapping) information: early wrap, late wrap, long-to-short wrap, and short-to-long wrap.

Early wrap This case has the best performance; it involves wrapping the exponents a cycle early and then wrapping the fraction into the E1, or execute-first, cycle as the data are being written to the register file. The exponent dataflow does not have late multiplexors in the E1 cycle, since they are timing-critical in the add-1 cycle. Instead, the exponent is wrapped back to the A and B exponent registers directly from the FS3, FS4, or FS6 state. This involves wrapping two separate exponents for a multiplication that is completing from the FS3 because of the different normalizations that are created on the fly. The following shows the relative timing of control signals and dataflow signals:

```
END OP        WR/E1
EXP WRAP    Fraction to late mux
```

END OP represents the end of the instruction handshake signal, EXP WRAP is the exponent wrapping, WR indicates the write cycle, and E1 the first cycle of execution. The wrapping takes two cycles.

Late wrap If the instruction that is the target of the bypass is not received early enough for the exponent to be wrapped early, the wrap is known as a late wrap. This involves wrapping the exponent and fraction during the write cycle to the A and B fraction and exponent registers. The following is the timing:

```
END OP    WR                     E1
        EXP/fraction to
         A and B registers
```

Long-to-short wrap For a long-to-short wrap, the timing is the same as for the late wrap. A longer data type is being written to the FPRs than has to be read from the FPRs. The low-order data must be masked, and there is no masking function on the late multiplexors. However, there is masking on the input to the A and B registers. Thus, the data must be wrapped to the input of the A and B registers. Another case of this wrap condition is creating a true zero (fraction, characteristic, and sign are set equal to zero) on a significance exception (result equal to zero). The true zeroing of the exponent for significance exception is performed in the FC3 exponent multiplexor. Thus, an early exponent wrap would have invalid data. The following is the timing:

```
END OP    WR                     E1
         EXP/Fraction to
         A and B registers
```

Short-to-long wrap For a short-to-long wrap, the data must be reread from the FPRs. There is no merge capability on the input to the A and B registers or in the late multiplexors. Thus, if data from a write must be combined with low-order data from the register file, a reread must take place. In addition to this case, exponent underflow, which results in a true zero, is detected very late and is bypassed in the same manner. The zeroing of the result for exponent underflow actually takes place in the C bus multiplexing of FPU_C_BUS and FXU_C_BUS prior to the register file. Thus, bypassing this exponent requires a reread of the register file; the timing is shown below:

```
END OP   WR   Read   E1
```

The early wrap is the most common of the wraps; it demonstrates an interesting design strategy. Methods of bypassing data can be divided into four categories:

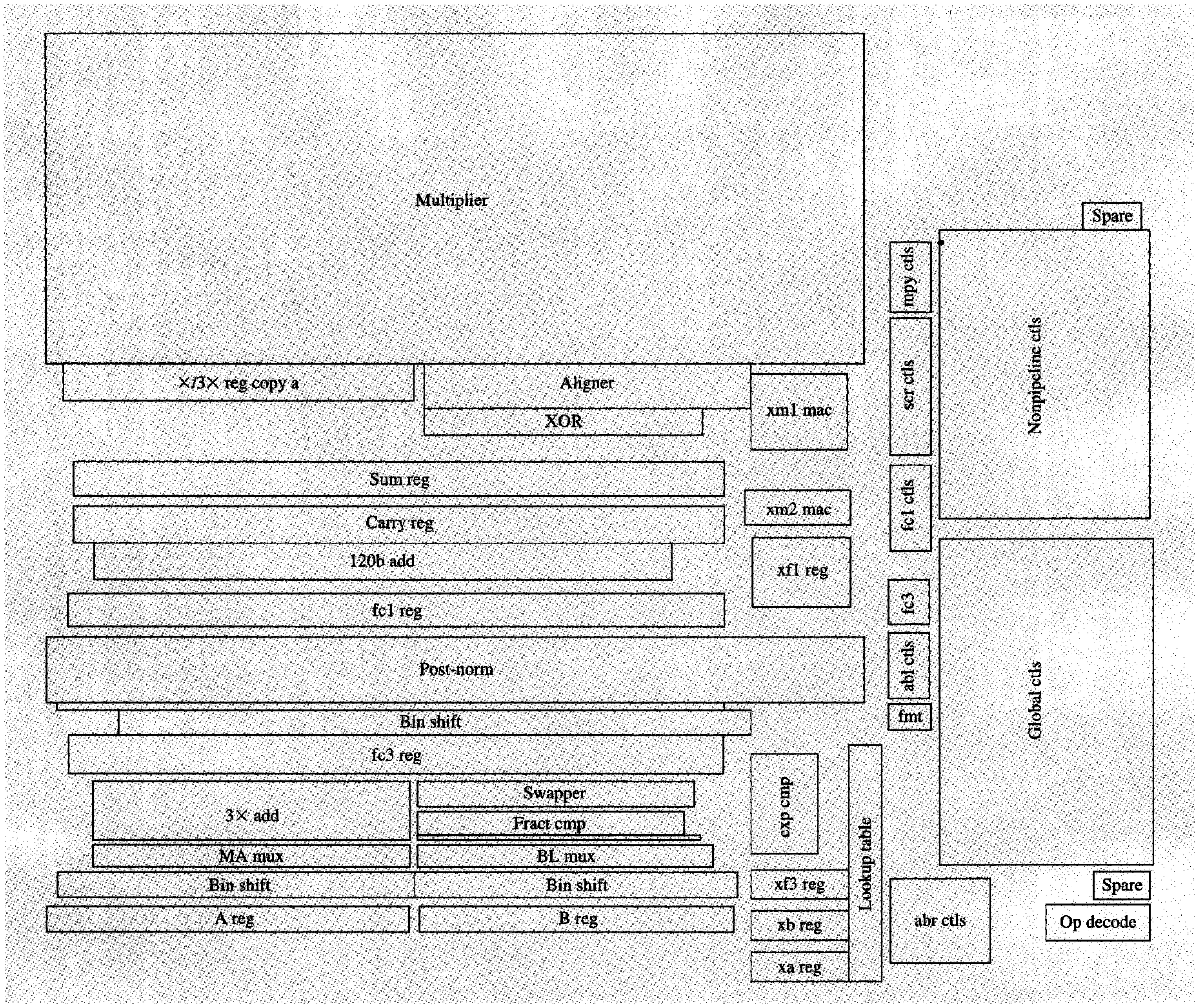


Figure 6

Macro layout.

1. Do not overlap the read and write cycle.
2. Overlap the read and write cycle.
3. Overlap the last execute cycle with the read cycle.
4. Overlap the write cycle with the first execute cycle.

The first two methods are low-performance and can be used for infrequent or complex cases. The last two methods are high-performance. However, method 4, which has been implemented for the early wrap case, has an interesting advantage in our implementation over method 3. The last execution stage can take place in several different pipeline stages: FS3, FS4, or even FS6. To implement method 3, a new bus or potentially several new buses would be needed to wrap the fraction data to the A and B registers. For method 4, only the C bus has to be used to drive into the late multiplexors. Since the C bus is an existing bus, no new tracks are needed for bypassing the fraction. Implementing method 4 reduces wiring congestion and gives good performance on data dependencies.

Physical design

The unit floorplan is shown in **Figures 6** and **7.** In effect, it is optimized around the multiplier. In order to meet the cycle time in the multiplier, we have reduced the 19 partial products generated in the fastest way possible—with six levels of CSA. This approach is very wire-intensive and consumes all of the wire resources on metal 1

to metal 4 in the heart of the multiply array. This fact was recognized early in the design, and the multiplier was placed at the very top of the unit (corner of the chip) to eliminate any wires that would have to cross it.

The dataflow was partitioned in 126-bit fraction and 14-bit exponent stacks. Some room on the sides of stacks was allocated for overflow logic required by many macros. All of the macros in the first cycle of execution are 60 bits wide (see Figures 1 and 2), while the rest of the macros are about 120 bits wide. To reduce unit area, we placed several 60-bit macros side by side. This created one horizontal wiring channel of 56 bits; however, overall we were able to reduce stack height.

Dataflow stacks were placed and wired manually. Since dataflow macro definitions were stable early in the design process, this approach led to the most compact design. Tracks were partitioned according to wire length, and wide wires were used for long nets to minimize delays and slews. Control macros were manually placed and wired with a vendor tool. A program was written that located areas for decoupling capacitor placements. The decoupling capacitors were placed both within and outside macros for a unit total of over 10 nF.

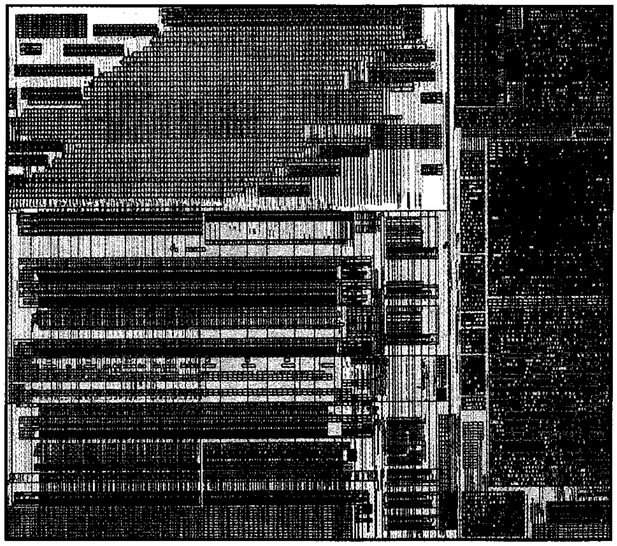


Figure 7

Physical design.

Circuit implementation

Most of the macros were implemented with static CMOS circuits. Only fraction/exponent dataflow registers and divide/square-root lookup ROS (read only storage) are dynamic [20]. The use of dynamic circuits was always an option for designers if delay goals were not met. However, through careful optimization of circuits in critical paths, we were able to meet our cycle-time objectives with what amounts to all static CMOS circuits.

Fraction/exponent dataflow macros were custom-designed. Even when common cells were used among several macros (such as register building blocks), they were custom-wired. Custom design was required to achieve both the cycle-time and area budgets in the dataflow macros. Control macros, on the other hand, were all synthesized and placed and routed with fixed-power library books. There are approximately 320K dataflow FETs in an area of 18.3 mm^2 and 61K control FETs in an area of 5 mm^2.

The control macros were implemented using a customized version of the IBM BooleDozer* logic synthesis tool. In order to meet the design schedule, it was very important to concentrate on improving our synthesis tool rather than manually tuning the results from synthesis. The design team restructured the VHDL to obtain improved synthesis implementation of the logic. This was done in parallel with focusing on improving the standard cell library and the logic transforms used within synthesis. The result of this effort was that all of the FPU control logic initially targeted for synthesis achieved the cycle-time goal and area constraints.

The timing reduction effort was facilitated by frequently running an FPU-level timing analysis to create timing reports and macro-level timing assertions used for synthesis and custom design. Unit-level timing runs were run in parallel with full-chip timing runs to create the necessary timing assertions. See [21] for more details on the design methodology.

Summary

A CMOS S/390 floating-point unit has been described which has been demonstrated at more than 400 MHz [22]. The design was optimized for frequently executed operations, and other instructions were implemented with cost-efficient algorithms. Some uncommon and aggressive algorithms were implemented (e.g., radix-8 multiplication; Goldschmidt division and square root); these algorithms were coupled with remainder-avoidance algorithms and parallel exponent calculation for normalization. Because the design caused critical control signals to be included in the dataflow, dataflow paths rather than control paths determined the cycle time. This made possible a reduced cycle time, which, coupled with a throughput of one cycle per instruction and a latency of three cycles for the most common additions and multiplications, resulted in a relatively high-performance FPU.

Acknowledgment

Design of the G4 floating-point unit would not have been possible without the combined efforts of the entire G4 project team. We especially acknowledge the contributions of Bob Averill, Robert Bunce, Rick Dennis, Dale Hoffman, Mike Mullen, Greg Northrop, Dianne Wassick, Charles Webb, Dave Webber, Barry Winter, Tom Wohlfahrt, and Fanchieh Yee.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. *Enterprise Systems Architecture/390*, Fourth Edition, Order No. SA22-7201-03, September 1996; available through IBM branch offices.
2. E. M. Schwarz, T. McPherson, and C. Krygowski, "Carry Select and Input Select Adder for Late Arriving Data," *Proceedings of the 30th Asilomar Conference on Signals, Systems, and Computers*, November 1996, pp. 182–185.
3. S. Vassiliadis, D. S. Lemon, and M. Putrino, "S/370 Sign-Magnitude Floating-Point Adder," *IEEE J. Solid-State Circuits* **24,** No. 4, 1062–1070 (August 1989).
4. S. Dao-Trong and K. Helwig, "A Single-Chip IBM System/390 Floating-Point Processor in CMOS," *IBM J. Res. Develop.* **36,** No. 4, 733–749 (July 1992).
5. S. Waser and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, CBS College Publishing, New York, 1982.
6. K. Hwang, *Computer Arithmetic: Principles, Architecture and Design*, John Wiley & Sons, Inc., New York, 1979.
7. R. M. Jessani and C. H. Olson, "The Floating-Point Unit of the PowerPC 603e Microprocessor," *IBM J. Res. Develop.* **40,** No. 5, 559–566 (September 1996).
8. S. Vassiliadis, E. M. Schwarz, and D. J. Hanrahan, "A General Proof for Overlapped Multiple-Bit Scanning Multiplications," *IEEE Trans. Comput.* **38,** No. 2, 172–183 (February 1989).
9. S. Vassiliadis, E. M. Schwarz, and B. M. Sung, "Hard-Wired Multipliers with Encoded Partial Products," *IEEE Trans. Comput.* **40,** No. 11, 1181–1197 (November 1991).
10. E. M. Schwarz, B. Averill, and L. Sigal, "A Radix-8 CMOS S/390 Multiplier," *Thirteenth Symposium on Computer Arithmetic*, Asilomar, CA, July 1997, pp. 2–9.
11. R. E. Goldschmidt, "Applications of Division by Convergence," Master's thesis, M.I.T., Cambridge, MA, June 1964.
12. S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers, "The IBM System/360 Model 91: Floating-Point Execution Unit," *IBM J. Res. Develop.* **11,** No. 1, 34–53 (January 1967).
13. M. J. Flynn, "On Division by Functional Iteration," *IEEE Trans. Comput.* **C-19,** No. 8, 702–706 (August 1970).
14. A. Svoboda, "An Algorithm for Division," *Information Processing Machines* **9,** 25–32 (1963); Prague, Czechoslovakia.
15. E. V. Krishnamurthy, "On Optimal Iterative Schemes for High-Speed Division," *IEEE Trans. Comput.* **C-19,** No. 3, 227–231 (March 1970).
16. M. Darley, B. Kronlage, D. Bural, B. Churchill, D. Pulling, P. Wang, R. Iwamoto, and L. Yang, "The TMS390C602A Floating-Point Coprocessor for Sparc Systems," *IEEE Micro* **10,** No. 3, 36–47 (June 1990).
17. E. M. Schwarz, "Rounding for Quadratically Converging Algorithms for Division and Square Root," *Proceedings of the 29th Asilomar Conference on Signals, Systems, and Computers*, October 1995, pp. 600–603.
18. C. V. Ramamoorthy, J. R. Goodman, and K. H. Kim, "Some Properties of Iterative Square-Rooting Methods Using High-Speed Multiplication," *IEEE Trans. Comput.* **C-21,** No. 8, 837–847 (August 1972).
19. S. Vassiliadis and E. M. Schwarz, "Controlling Unit for a Pipelined Floating Point Hard-Wired Engine," *Proceedings of the IFIP Third International Workshop on Wafer Scale Integration*, June 1989, pp. 343–351.
20. L. Sigal, J. D. Warnock, B. W. Curran, Y. H. Chan, P. J. Camporese, M. D. Mayo, W. V. Huott, D. R. Knebel, C. T. Chuang, J. P. Eckhardt, and P. T. Wu, "Circuit Design Techniques for the High-Performance CMOS IBM S/390 Parallel Enterprise Server G4 Microprocessor," *IBM J. Res. Develop.* **41,** No. 4/5, 489–503 (1997, this issue).
21. K. L. Shepard, S. M. Carey, E. K. Cho, B. W. Curran, R. F. Hatch, D. E. Hoffman, S. A. McCabe, G. A. Northrop, and R. Seigler, "Design Methodology for the S/390 Parallel Enterprise Server G4 Microprocessors," *IBM J. Res. Develop.* **41,** No. 4/5, 515–547 (1997, this issue).
22. C. Webb, C. Anderson, L. Sigal, K. Shepard, J. Liptay, J. Warnock, B. Curran, B. Krumm, M. Mayo, P. Camporese, E. Schwarz, M. Farrell, P. Restle, R. Averill, T. Slegel, W. Huott, Y. Chan, B. Wile, P. Emma, D. Beece, C. Chuang, and C. Price, "A 400 MHz S/390 Microprocessor," *ISSCC Digest of Technical Papers*, pp. 168–169 (February 1997).

Received December 5, 1996; accepted for publication April 17, 1997

Eric M. Schwarz *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (ESCHWARZ at PK705VMA, schwarz@vnet.ibm.com).* Dr. Schwarz received a B.S. degree in engineering science from The Pennsylvania State University in 1983, an M.S. degree in electrical engineering from Ohio University in 1984, and a Ph.D. degree in electrical engineering from Stanford University in 1993. He joined IBM in 1984 in Endicott, New York, and in 1993 transferred to Poughkeepsie. Dr. Schwarz is an Advisory Engineer and was FPU Logic Technical Leader for the S/390 Parallel Enterprise Server G4 processor. Currently, he is Execution Unit (FPU and FXU) Logic Technical Leader for follow-on processors. His research interests are in computer arithmetic and computer architecture. He is the author of seven filed patents, ten pending patents, and several journal articles and conference proceedings.

Leon Sigal *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (LJS at YKTVMV, ljs@vnet.ibm.com).* Mr. Sigal received a B.S. in biomedical engineering in 1985 from the University of Iowa and an M.S. in electrical engineering in 1986 from the University of Wisconsin at Madison. He worked at Hewlett-Packard's microprocessor development laboratory between 1986 and 1992. Mr. Sigal joined IBM in 1992 and has been leading the CMOS S/390 microprocessor circuit design interdivisional effort.

Thomas J. McPherson *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (MCPHERSO at PK705VMA, tmcpherson@vnet.ibm.com).* Mr. McPherson is a Staff Engineer in S/390 microprocessor development. He received a B.S. degree in electrical engineering from Rutgers University in 1990 and an M.S. degree in computer engineering from Syracuse University in 1992. Mr. McPherson joined IBM in 1990 and has worked on S/390 microprocessors and CMOS ASIC designs.

08/414,250

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

SERIAL NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NO.
08/414,250	03/31/95	SCHWARZ E	PD994-050

24M1/0411

CHRISTOPHER A HUGHES
MORGAN AND FINNEGAN
345 PARK AVENUE
NEW YORK NY 10154

EXAMINER	
NGO, C	
ART UNIT	PAPER NUMBER
2306	8

04/11/97

DATE MAILED:

NOTICE OF ALLOWABILITY

PART I.

1. ☑ This communication is responsive to papers filed on 12/11/96.
2. ☑ All the claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice Of Allowance And Issue Fee Due or other appropriate communication will be sent in due course
3. ☑ The allowed claims are 1-17.
4. ☐ The drawings filed on ______ are acceptable.
5. ☐ Acknowledgment is made of the claim for priority under 35 U.S.C. 119. The certified copy has [_] been received [_] not been received. [_] been filed in parent application Serial No ______, filed on ______.
6. ☐ Note the attached Examiner's Amendment.
7. ☐ Note the attached Examiner Interview Summary Record, PTOL-413.
8. ☑ Note the attached Examiner's Statement of Reasons for Allowance.
9. ☐ Note the attached NOTICE OF REFERENCES CITED, PTO-892.
10. ☐ Note the attached INFORMATION DISCLOSURE CITATION, PTO-1449.

PART II.

A SHORTENED STATUTORY PERIOD FOR RESPONSE to comply with the requirements noted below is set to EXPIRE THREE MONTHS FROM THE "DATE MAILED" indicated on this form Failure to timely comply will result in the ABANDONMENT of this application Extensions of time may be obtained under the provisions of 37 CFR 1 136(a)

1. ☐ Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL APPLICATION, PTO-152, which discloses that the oath or declaration is deficient A SUBSTITUTE OATH OR DECLARATION IS REQUIRED
2. ☑ APPLICANT MUST MAKE THE DRAWING CHANGES INDICATED BELOW IN THE MANNER SET FORTH ON THE REVERSE SIDE OF THIS PAPER
   a. ☑ Drawing informalities are indicated on the NOTICE RE PATENT DRAWINGS, PTO-948, attached hereto or to Paper No. 5 CORRECTION IS REQUIRED.
   b ☐ The proposed drawing correction filed on ______ has been approved by the examiner. CORRECTION IS REQUIRED.
   c ☐ Approved drawing corrections are described by the examiner in the attached EXAMINER'S AMENDMENT CORRECTION IS REQUIRED.
   d ☑ Formal drawings are now REQUIRED

Any response to this letter should include in the upper right hand corner, the following information from the NOTICE OF ALLOWANCE AND ISSUE FEE DUE: ISSUE BATCH NUMBER, DATE OF THE NOTICE OF ALLOWANCE, AND SERIAL NUMBER.

Attachments:

- _ Examiner's Amendment
- _ Examiner Interview Summary Record PTOL-413
- √ Reasons for Allowance
- _ Notice of References Cited, PTO-892
- _ Information Disclosure Citation, PTO-1449
- _ Notice of Informal Application, PTO-152
- _ Notice re Patent Drawings, PTO-948
- _ Listing of Bonded Draftsmen
- _ Other

Chuong Ngo

CHUONG D. NGO
PATENT EXAMINER
GROUP 2300

PTOL-37 (REV. 4-89) *

Serial No. 08/414,250 2 Attachment to paper No. 8
Art Unit 2306

REASONS FOR ALLOWANCE

1. The following is an Examiner's Statement of Reasons for Allowance:

The prior art of record does not fairly suggest in a computer system a conversion means as recited in claims 1, and the first through forth converter as recited in claim 11 wherein, in accordance with 35 USC 112, 6th paragraph, the conversion means is construed to cover the corresponding structure of the combination of converters 11,12,15,16,22,24, multiplexers 28,28,29 and register 13,14,14 connected as disclosed in figure 1 or equivalents thereof, and the first and forth converter apply the same transformation for both converting an operand of a first floating point architecture type to the internal floating point format, and transforming an operand of a second floating point architecture type as recited in claim 11.

2. Any comments considered necessary by applicant must be submitted no later than the payment of the Issue Fee and, to avoid processing delays, should preferably **accompany** the Issue Fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

3. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chuong D. Ngo whose telephone number is (703) 305-9764.

Serial No. 08/414,250 3 Attachment to paper No. 8
Art Unit 2306

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3800

The fax number for this Group is (703) 305-9724.

Chuong D. Ngo
Primary Examiner
Art Unit 2306

04007-97

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

Address: Box ISSUE FEE
ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

NOTICE OF ALLOWANCE AND ISSUE FEE DUE

24M1/0411

CHRISTOPHER A HUGHES
MORGAN AND FINNEGAN
345 PARK AVENUE
NEW YORK NY 10154

APPLICATION NO.	FILING DATE	TOTAL CLAIMS	EXAMINER AND GROUP ART UNIT		DATE MAILED
08/414,250	03/31/95	017	NGO, C	2306	04/11/97

First Named Applicant: SCHWARZ, ERIC M.

TITLE OF INVENTION: IMPLEMENTATION OF BINARY FLOATING POINT USING HEXADECIMAL FLOATING POINT UNIT

ATTY'S DOCKET NO.	CLASS-SUBCLASS	BATCH NO.	APPLN. TYPE	SMALL ENTITY	FEE DUE	DATE DUE
3 P0994-050	364-748.000	L75	UTILITY	NO	$1290.00	07/11/97

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED.

THE ISSUE FEE MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED.

HOW TO RESPOND TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.
If the SMALL ENTITY is shown as yes, verify your current SMALL ENTITY status:

A. If the status is changed, pay twice the amount of the FEE DUE shown and notify the Patent and Trademark Office of the change in status, or
B. If the status is the same, pay the FEE DUE shown above.

If the SMALL ENTITY is shown as NO:

A. Pay FEE DUE shown above, or

B. File verified statementof Small Entity Status before, or with, payment of 1/2 the FEE DUE shown above.

II. Part B of this notice should be completed and returned to the Patent and Trademark Office (PTO) with your ISSUE FEE. Even if the ISSUE FEE has already been paid by charge to deposit account, Part B should be completed and returned. If you are charging the ISSUE FEE to your deposit account, section "6b" of Part B should be completed.

III. All communications regarding this application must give application number and batch number. Please direct all communication prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

A single-chip IBM System/390 floating-point processor in CMOS

by S. Dao-Trong
K. Helwig

A floating-point processor with the IBM System/390® architecture is implemented in one CMOS VLSI chip containing over 70 000 cells (equivalent inverters), using a transistor channel length of 0.5 μm. All floating-point instructions are hard-wired, including the binary integer multiplications. The chip is implemented in a 1-μm technology with three layers of metal. All circuits are realized in standard cells except for a floating-point register and a multiplier array macro, which are custom designed to save chip area. Instructions are performed in a five-stage pipeline with a maximum operating frequency of 37 MHz. The chip measures 12.7 mm × 12.7 mm, and dissipates 2 W. It is part of the chip set which forms the core of the IBM Enterprise System/9000™ Type 9221 entry-level models.

Introduction

Floating-point processors used to be seen as an option which could be added to the main CPU for performing scientific applications. In the models of the entry-level type (designated 9221) of the new IBM Enterprise System/9000™ (ES/9000™) line, floating-point processing is becoming an inherent part of the CPU. This paper describes the IBM ES/9000 Type 9221 floating-point processor, which is tightly coupled to the CPU and carries out all IBM System/390® floating-point instructions. All instructions are hardware-coded, so no microinstructions are needed. Moreover, binary integer multiplication is also implemented on the floating-point unit to improve overall performance.

The floating-point processor was implemented in 1-μm CMOS technology, using the standard cell approach, with only two custom-designed macros—the 60 × 58-bit multiplier macro and the floating-point register macro. The data flow design was done in a top-down manner, starting with an abstract functional block description which was broken down in more and more detail until the gate implementation level was reached. This method is especially appropriate for a standard cell approach, in which only logic gates from a standard library are available and no detailed knowledge about the transistor level is required. A mixed-level representation was created which was used for understanding, documenting, and most of all, early floor-planning, to make sure that all the logic fit on the chip and that the wiring delays were acceptable.

Design goals

Because of the inherent character of the floating-point processor in the IBM Enterprise System/9000 Type 9221 models, a global optimization of the processor chip set was the ultimate design goal. First, system partitioning with

respect to packaging requirements was done. Our overall goal was to obtain the maximum system performance with the given CMOS process [1]. Wide buses were used to achieve high performance. Critical interconnections were split into an internal processor bus and a floating-point bus. The chip set thus built consisted of two multichip modules, one containing the fixed-point unit, the instruction processing unit, and two cache chips, and the other with only the floating-point processor unit and the clock chip. See [1] for additional details.

Second, local chip optimization was done. For the floating-point unit, the cycle was dictated by the CPU. Optimization was then focused on pipeline structure, chip area, and instruction implementation. Much effort was expended to make sure that the structure would fit on one chip. Although there were many implementation alternatives with more promising cycle times, we chose one which best fit the overall requirements of chip area and cycle time.

Chip structure

• *Pipelining*

While the CPU is based on a four-stage pipeline, the floating-point processor requires a five-stage pipeline to perform its most used instructions (add, subtract, and multiply) in one cycle for double-precision operands. The CPU resolves operand addresses, provides operands from the cache, and handles all exceptions for the floating-point processor. The five stages of the pipeline are instruction fetch, which is executed on the CPU, register fetch, operand prealignment, addition, and normalization and register store.

To preserve synchronization with the CPU, a floating-point wait signal is raised whenever a floating-point instruction needs more than one cycle. The CPU then waits until this wait signal disappears before it increments its program counter and starts the next serial instruction, which is kept on the bus.

Because the IBM System/390 architecture requires that interrupts be precise, a wait condition is also invoked whenever an exception may occur. To minimize the impact on performance, extra logic was incorporated to predetermine the interrupt situations, as described later in the discussion of control flow.

• *Data flow*

The data flow of the IBM ES/9000 Type 9221 floating-point processor was designed in a top-down manner. Starting with global function blocks, we broke down the next hierarchical level into more detailed functions that were finally transformed into gate-level functions, including fine tuning with respect to logic transformation, input assignment, and signal buffering. The last is very critical for CMOS technology.

Figure 1 shows the data flow of the IBM ES/9000 Type 9221 floating-point processor. It has been designed to perform the most used add/subtract and multiply instructions in one cycle for single- and double-precision operands. Many bypass buses are used to avoid wait cycles when the results of the foregoing instructions are used. A wait cycle is needed only if the result of one instruction is used immediately by the next sequential instruction.

The data flow shows two parallel paths for fraction processing: one add-path where all non-multiply instructions are implemented, and one multiply-path especially designed for multiply and divide. The add-path is 60 bits wide and consists of an operand switcher, an aligner, an adder, and a normalizer shifter. Instead of using two aligners on each side of the operand paths, we used a switcher to switch operands, thereby saving one aligner. The switcher is also needed for other instructions, and requires many fewer circuits. The resulting delay is not critical.

The multiply-path consists of a Booth encoder for the 58-bit multiplier, a multiplier macro which forms the 58×60-bit product terms *sum* and *carry*, and a 92-bit adder which delivers the result product. The sign and exponent paths are adjusted to be consistent with the add-path. The exponent path resolves all exception and true zero situations, as defined by the IBM System/390 architecture. The implementation of all other instructions is merged into the add-path and multiply-path, and requires only minimal additional logic. The data flow in Figure 1 thus shows more function blocks and multiplexer stages than needed for only add, subtract, and multiply operations.

The data flow was then partitioned into smaller parts (typically registers with their input control). For every partition, such as FB, we described the functions in bit-level detail using a block description form to reflect all functions required. All interface signals were named appropriately. **Figure 2** shows an example of the detailed documentation for the partition FB. This was used for further transformation into gate-level and high-level documentation of the data flow, and also to design the control flow. We were able to stabilize the interfaces between partitions at a very early stage. Cell count prediction was then easily made, so that early floor-planning could be done effectively. Because of our numerous wide internal buses, the chip layout was based exclusively on wiring studies. (It was not obvious that the data flow would fit on the chip.) Using this high-level documentation, we obtained a chip floor plan at a very early stage. It helped us to adjust nets and bus buffering to achieve a well-balanced design. In fact, it was found that delay problems arise primarily in the long buses. Also, chip floor-planning ensured that we could implement all of

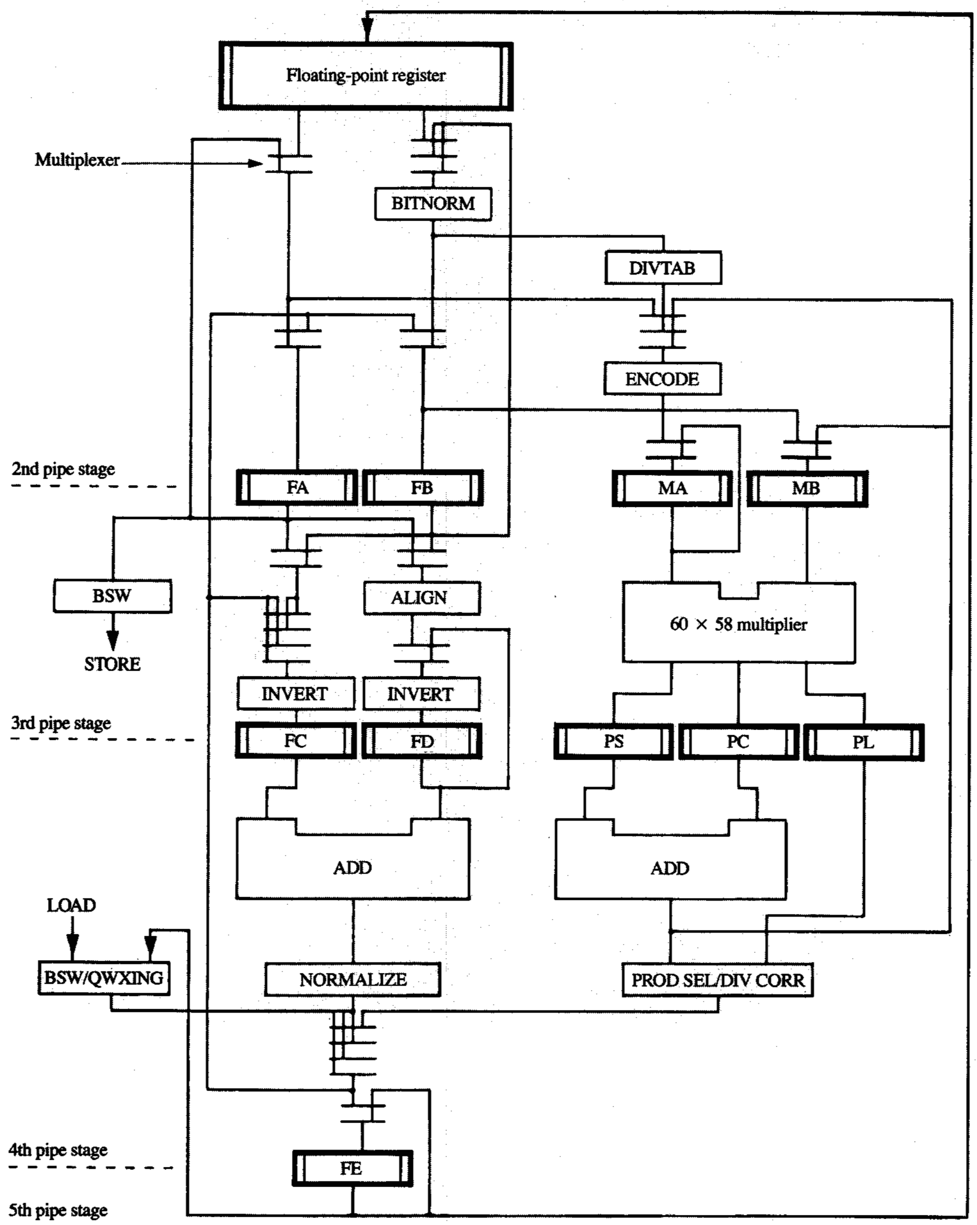


Figure 1

Floating-point data flow.

our logic on a single chip. **Figure 3** shows the chip partition and the cell count prediction for each partition. The main buses are globally wired. The control logic part was assumed to comprise about 20% of the data flow part.

Arithmetic implementation

Floating-point instructions are partitioned into three main groups: 1) addition/subtraction, load; 2) multiplication; and 3) division. These are the instructions most used in

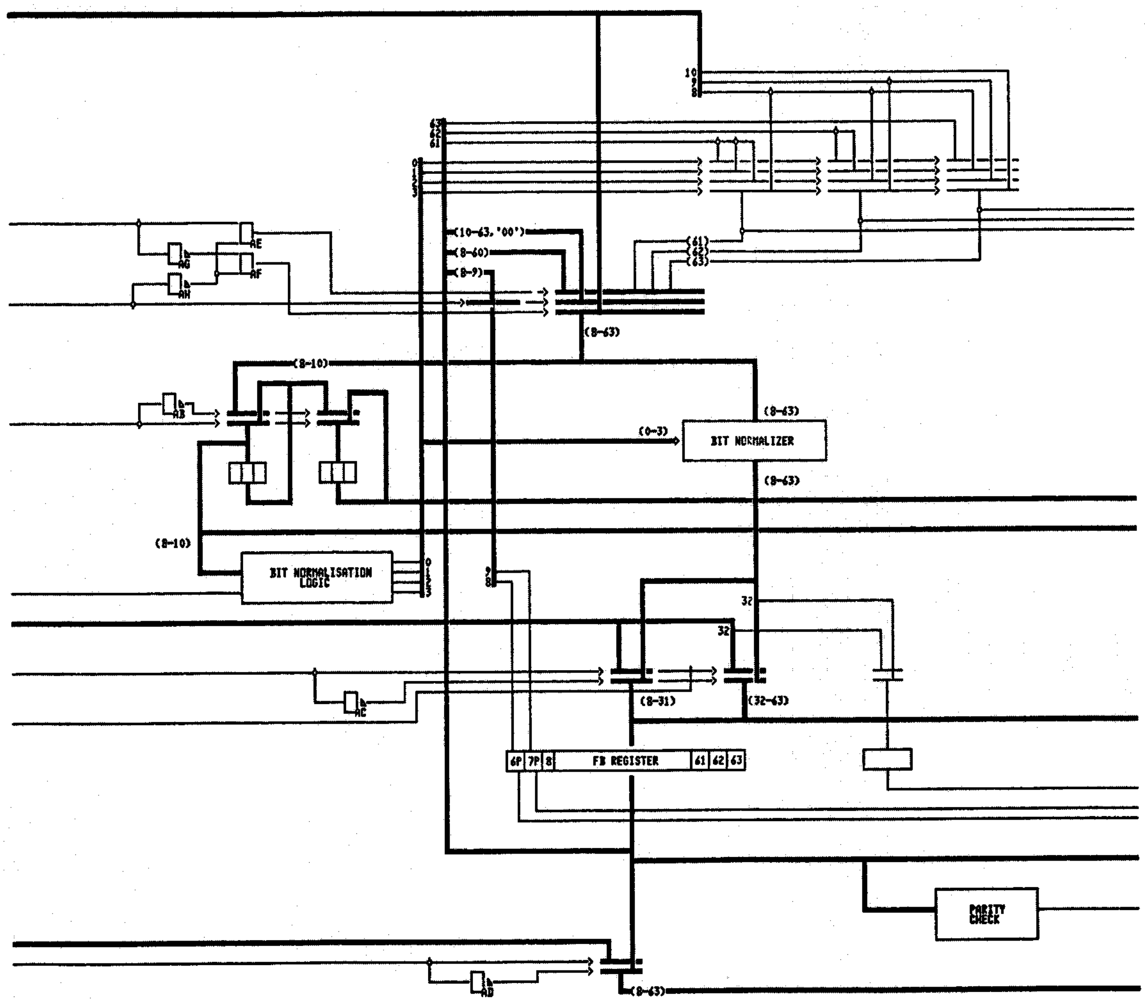


Figure 2

Detailed documentation of partition FB.

scientific applications. The first two groups of instructions are performed in one cycle, and division is made as fast as possible. The following subsections describe the control flow to these three main groups of instructions.

• *Addition, subtraction, load*

During the first two pipelined stages, only instruction and operand fetching are done. All data processing is concentrated in the third and fourth pipelined stages. In the fifth stage, the result is written back to the floating-point register. Hardware implementation is done with minimal logic, provided the cycle remains within the design range.

Effective addition, load

Loads are treated like addition, with one operand equal to zero. During stage 3 the exponents of both operands are compared in order to determine the amount of alignment shift. The operand with the smaller exponent is then passed to the aligner for prealignment. In stage 4 the aligned operands are added. The addition may produce a carry-out, which results in a shift right by one digit position, in accordance with the IBM System/390 architecture. The exponent must then be incremented by one. If the instruction requires postnormalization, leading zeros are detected, and the normalization shift amount is determined. The exponent is then decreased accordingly.

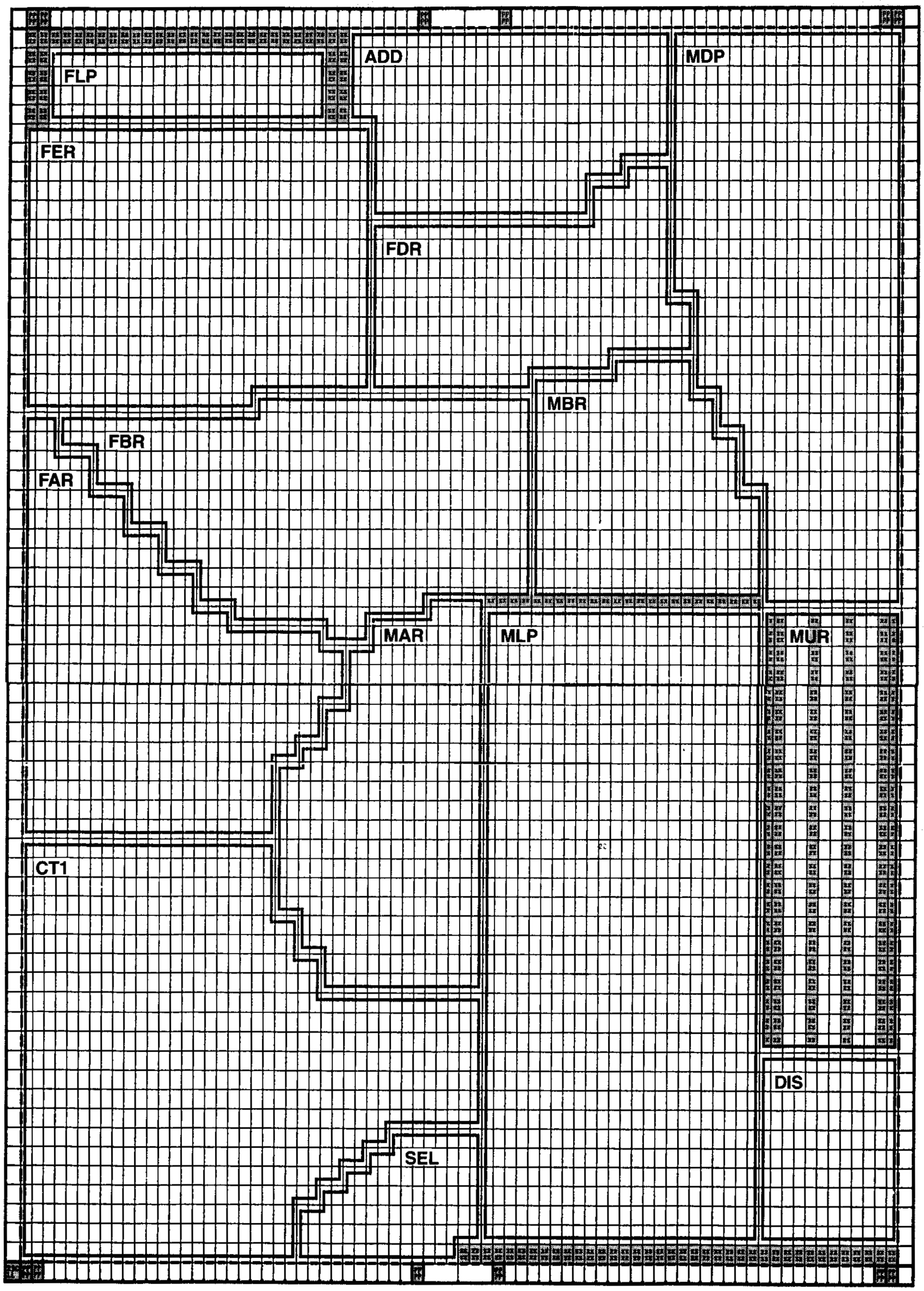


Figure 3

Early floor-planning. Areas PP are reserved pads; areas XX are reserved for wiring. From upper left to lower right, IDs of the areas are as follows: FLPT (floating-point register macro), ADD (adder), MDP (multiplier postprocessing), FER (FE partition), FDR (FD partition), MBR (MB partition), FAR (FA partition), FBR (FB partition), MAR (MA partition), MLP (multiplier macro), MUR (multiplier register), CT1 (control logic block), DIS (oscillator), and SEL (self-test logic).

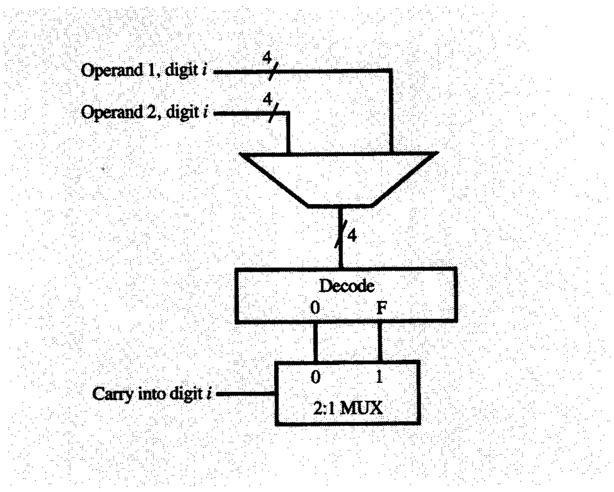


Figure 4

Zero-digit detection.

Table 1 Exception wait situations.

Effective addition/subtraction	*Exponent = 7F*	*Exponent < 0D*	*Unnormalized operand*
Yes	Yes	—	No
Yes	—	Yes	Yes
No	—	Yes	—

Since time is still available in stage 4, the exponent calculation is made sequentially after that of addition, using only one exponent adder with an input multiplexer to select whether an exponent increase, an exponent adjustment, or a multiply/divide exponent is required.

Leading-zero detection is made by calculating the hexadecimal digit sums without a propagated carry-in. Hexadecimal sums 0 and F for the digit position i are determined and fed into a multiplexer. The carry-in to this digit position selects whether or not the result digit is zero. This carry bit comes from the same carry-lookahead circuit used for the adder, so no additional circuit is needed. **Figure 4** shows the realization. By using this additional logic, the shift amount can be determined at nearly the same time as the addition result.

Exponent exception, either overflow or underflow, is also detected in this stage. Meanwhile, the next instruction has already been started. As mentioned earlier, a wait may be raised at stage 3 to hold execution of the next serial instruction. In our case of an effective addition, the wait situation is met when

- The intermediate result exponent is 7F and will overflow when an exponent increment is caused by a carry-out from the adder.
- The intermediate result exponent is smaller than 0D, and a normalization is required for unnormalized operands. Here the exponent must be decreased by the normalization shift amount, which can be at most 0D (14 in the decimal system), thus producing an exponent underflow.

Table 1 shows the cases in which an exponent exception condition is met for the effective addition and subtraction. This represents a very small percentage of cases, so performance is only minimally affected.

Effective subtraction

To avoid recomplementation of the result, we always subtract the smaller operand from the greater one. Most frequently, subtractions are done by means of normalized fractions. Here the greater operand can always be determined whenever the exponents are unequal. In stage 3 the operand with the smaller exponent is passed to the aligner whose output is to be complemented. When the exponents are equal, a fraction-compare circuit compares the operands and selects the smaller operand to be negated. Thus, we have a complemented block on both sides of stage 3 in our data flow. In stage 4 the subtraction is performed, delivering a positive result. The sign is the sign of the greater operand. Normalization is done in the same way as for an effective addition. No exponent overflow can occur.

For the case of subtraction with unnormalized operands, except for the case in which the exponents are equal, we cannot determine which operand is the smaller one. One additional cycle is needed. At first we calculate $A - B$. Depending on whether the result is positive or negative, we calculate in the following cycle $A - B$ or $B - A$, respectively. The result is then taken from the second subtraction. The sign is determined accordingly.

- *Multiplication*

Multiplication is implemented by using a multiplier macro custom-designed to save chip area. Because the overall design goal was not to minimize cycle time but to optimize the global chip set, we used the modified Booth algorithm with serial addition of partial product terms [2], although the Wallace tree method would have been more efficient [3]. The latter method, however, is not appropriate for a regular structure, which in turn is very important when custom design is required and chip area and turnaround time are to be minimized. When this method is used, the multiplication time is proportional to the number of additions of partial products, which is $N/2$, where N is the number of multiplication bits.

The design goal was to increase the arithmetic speed by reducing the number of additions and sequential delay

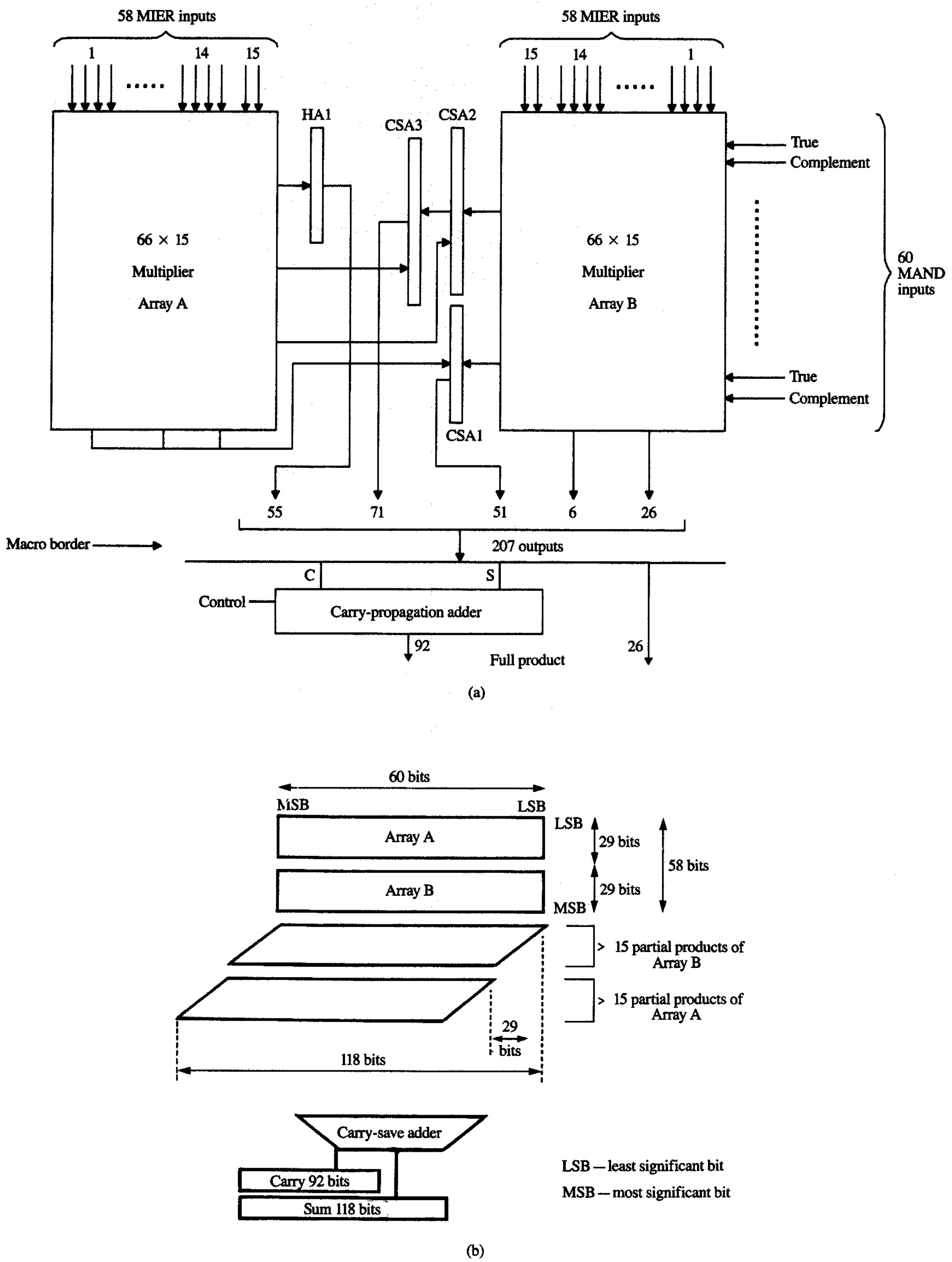


Figure 5

Multiply array: (a) physical partitioning; (b) logical description. Part (a) reprinted from [1] with permission; © 1990 IEEE.

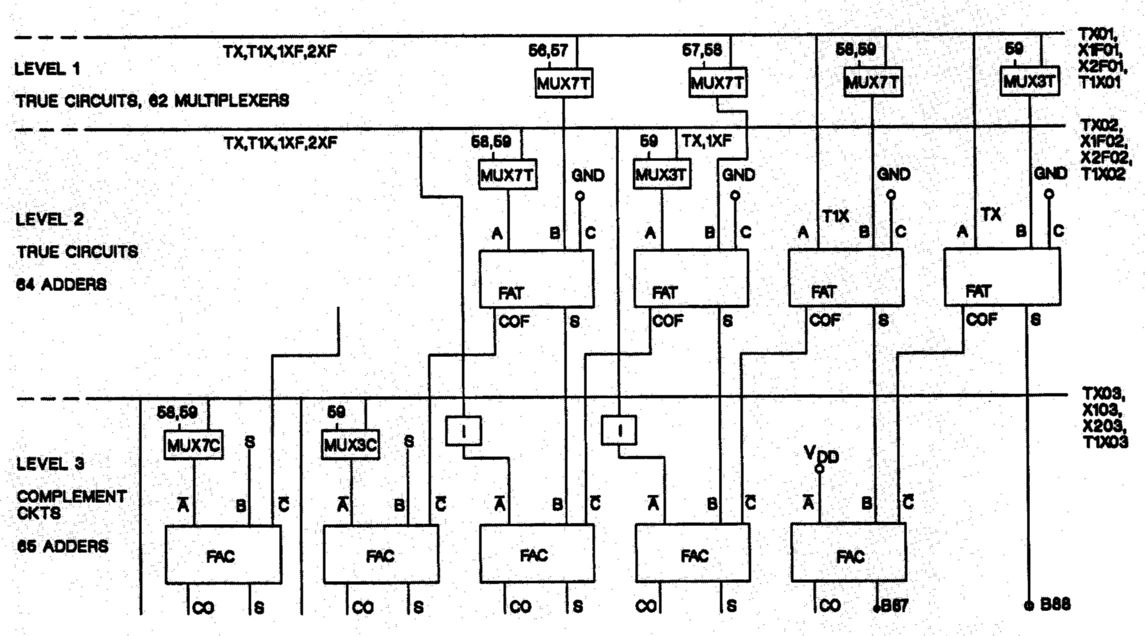


Figure 6

Multiplier array, upper corner.

stages. The modified Booth method is used to meet the first goal, and a partitioning of the multiplier array into two identical half parts to meet the latter goal.

Partitioning the array into two parts is not trivial. A special carry network has been developed to move the carry bits out of the lower significant array properly [4]. The advantages are the following: First, the carry network reduces the total multiplier delay by nearly 50%; second, it favors the structural regularity of the multiplier macro cell, as does the Booth algorithm. This regularity has the following advantages:

- The custom design is easier to handle (e.g., checking is easier).
- Only one array need be designed.
- The performance is enhanced, and the area used is reduced, by employing repeated circuits and repeated interconnection wiring. Circuit design and layout can be made very dense.
- The multiplier bit width is easily extendable.
- Regularity also means short wiring, which is a major factor affecting the performance of CMOS designs.
- Design turnaround time is reduced.

In the Type 9221 floating-point processor, the multiplier must support 60 × 58-bit multiplication, which is required for our division algorithm. A block diagram of the 60 × 58 multiplier is shown in **Figure 5**. The border of the macro cell is indicated by a dashed line. The registers at the inputs as well as at the outputs of the macro cell are part of the automatically wired logic circuitry of the chip. Inside the macro cell there are no latches, only multiplexers and adders.

The multiplier (MIER) input and output registers and the multiplicand (MAND) input registers act as input and output ports to the array. These registers are separate scan paths which enhance the testability of the multiplier. The decoders for the 2 × 29 multiplier inputs are placed before the multiplier input registers, to keep the delay of the multiplier and the rest of the data flow well balanced. The multiplier output registers are inserted before the final summation of the most significant product terms. This location was selected on the basis of the delay characteristics of the macrocell versus the cycle time of the system. Carry-save adders (CSA1, CSA2, and CSA3) and half adders (HA1) between Array A and Array B are used to combine the sums and carries coming from both arrays. Arrays A and B are mirrored on the chip, permitting all array bits to be concentrated in the middle channel for summation. The multiplicand inputs cross Array B, the carry-save adders, the half adders, and finally Array A, in straight lines.

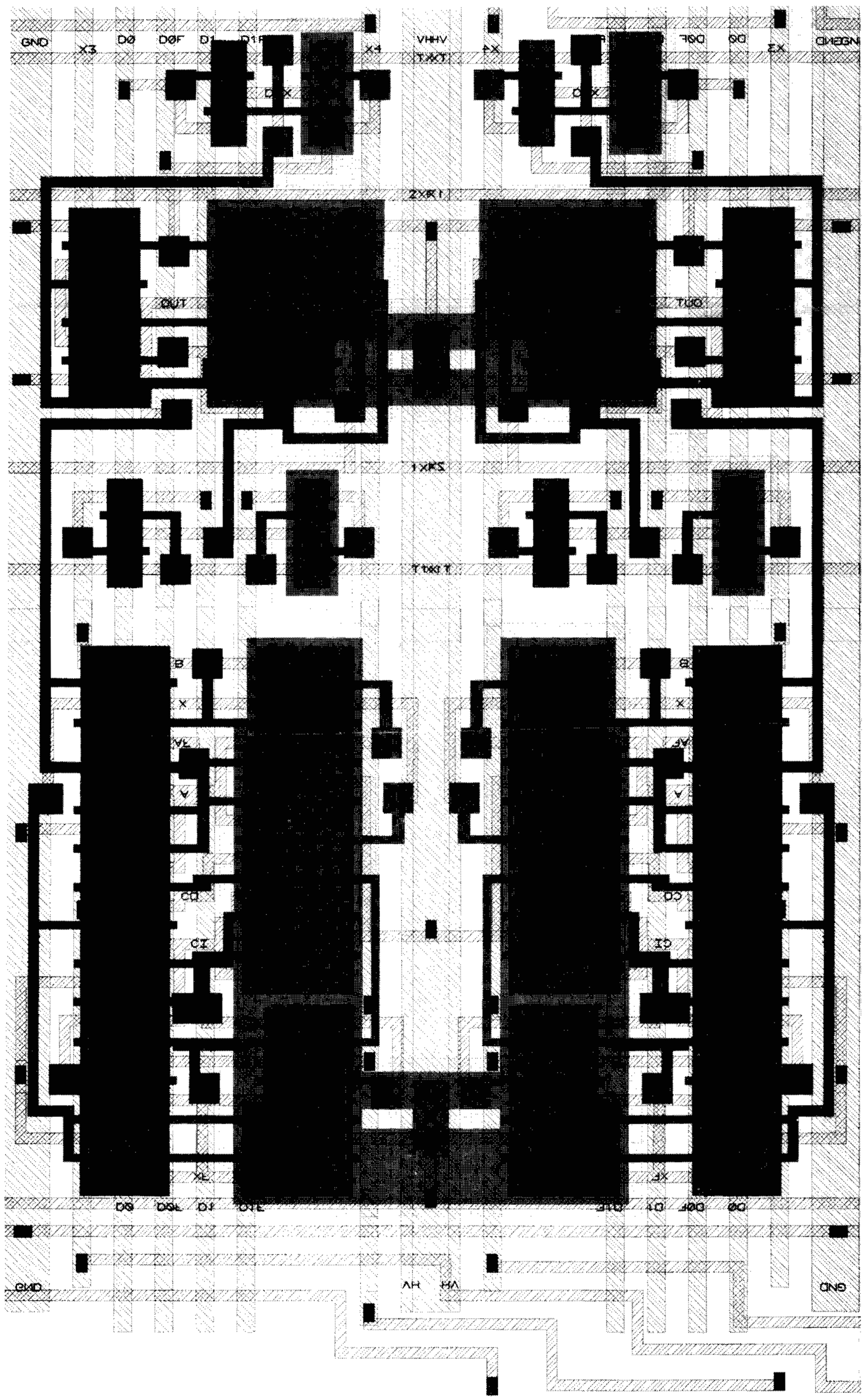


Figure 7

Mirrored pair of array elements.

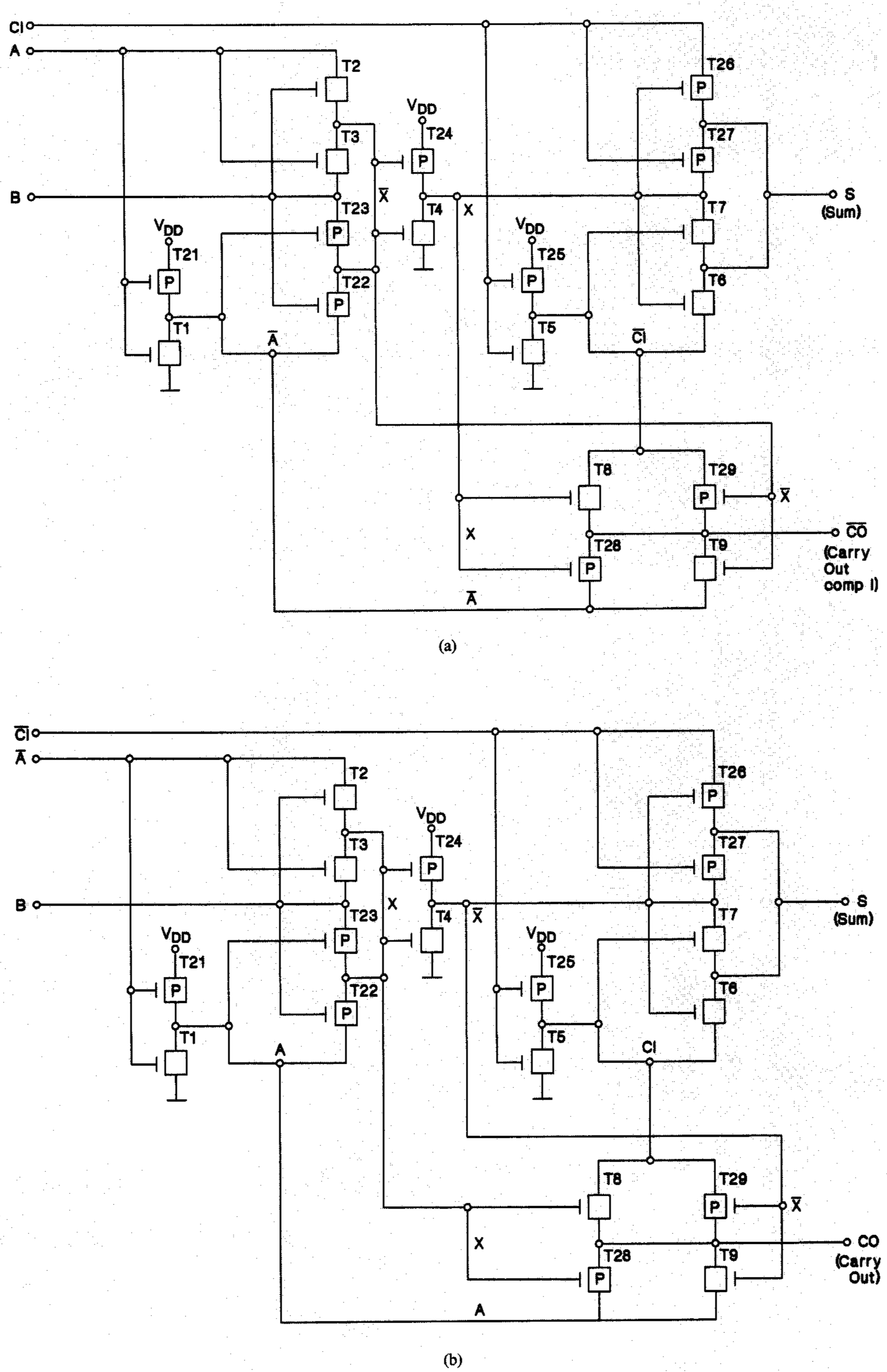


Figure 8

Adder with (a) true inputs and (b) complemented inputs. Transistors are n-type unless marked P for p-type.

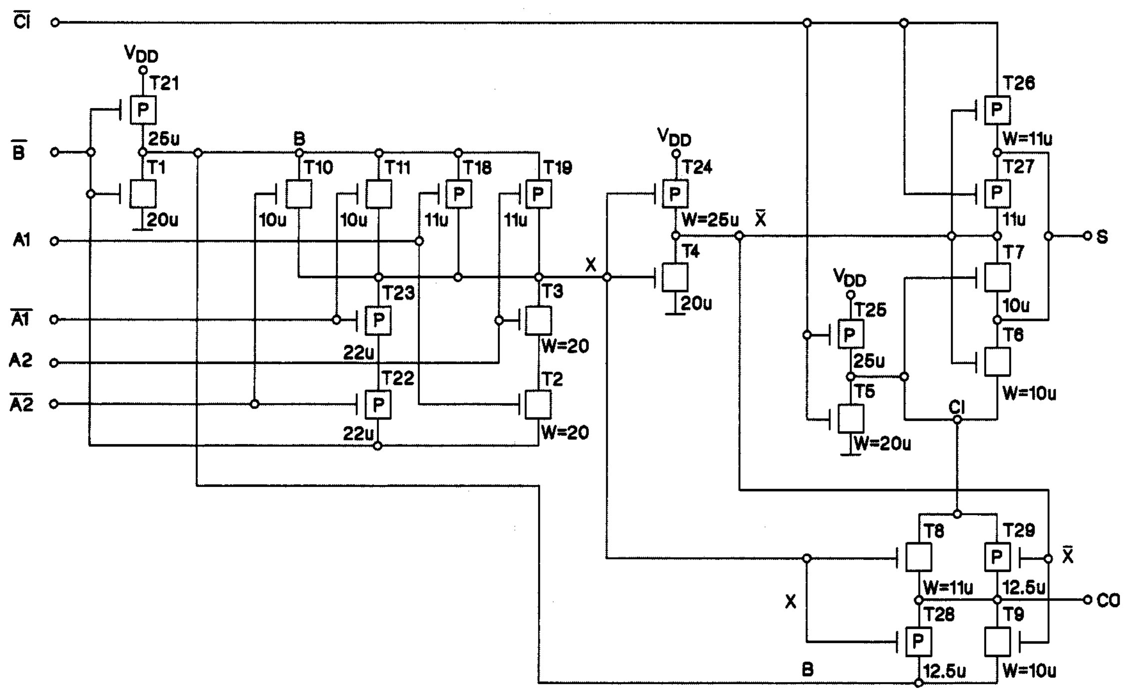


Figure 9

AND circuit integrated into the full adder. Transistors are n-type unless marked P for p-type.

To summarize, the multiplier uses the modified Booth algorithm, and consists of two identical arrays with 15 rows and 66 columns each. The first row contains only multiplexers for the selection of the Booth-encoded signals. Rows 2 through 15 contain multiplexers and full adders (carry-save adders).

Circuit organization

The structure of the multiplier arrays is partly shown in **Figure 6**, which contains the upper right corner of one array. The first row (Level 1) consists only of multiplexers (MUX7T, MUX3T). Full adders (FAT, FAC) are placed from row 2 on. To save inverters (and delay time), the circuits in row 2 are different from the circuits in row 3, a pattern that is repeated throughout the remainder of the array. The full adders in row 2 have only true input signals, a complemented carry output, and a true sum output. The full adders in row 3 have two complemented input signals, while the other input and output signals are true.

Signals 56 through 59 represent the last four (low-order) multiplicand input signals. These signals are inputs to the multiplexers. The four horizontal lines (TX01, · · ·) are Booth-encoded signals leading to all multiplexers of one row. In the macrocell layout, the circuits of a row are shifted by two pitches to the right compared with the circuits of the next higher row. This guarantees a rectangular array layout. There are different types of multiplexers and full adders derived from one multiplexer layout or from one full adder layout. One unique array element was optimally designed, with many personalization possibilities. It can be used on every row of the array, repeatedly. **Figure 7** shows the layout of a mirrored pair of array elements which makes the placement of the circuits in the array much simpler. Only at the right edge of the array are there irregularities. The full adder with an integrated AND-function was designed with special care from both layout and performance viewpoints, since it contributed to the slowest delay path.

Circuit description

The delay characteristics of the full adders used in the multiplier arrays directly affect the performance of the macrocell. Both the sum and the carry paths of the full

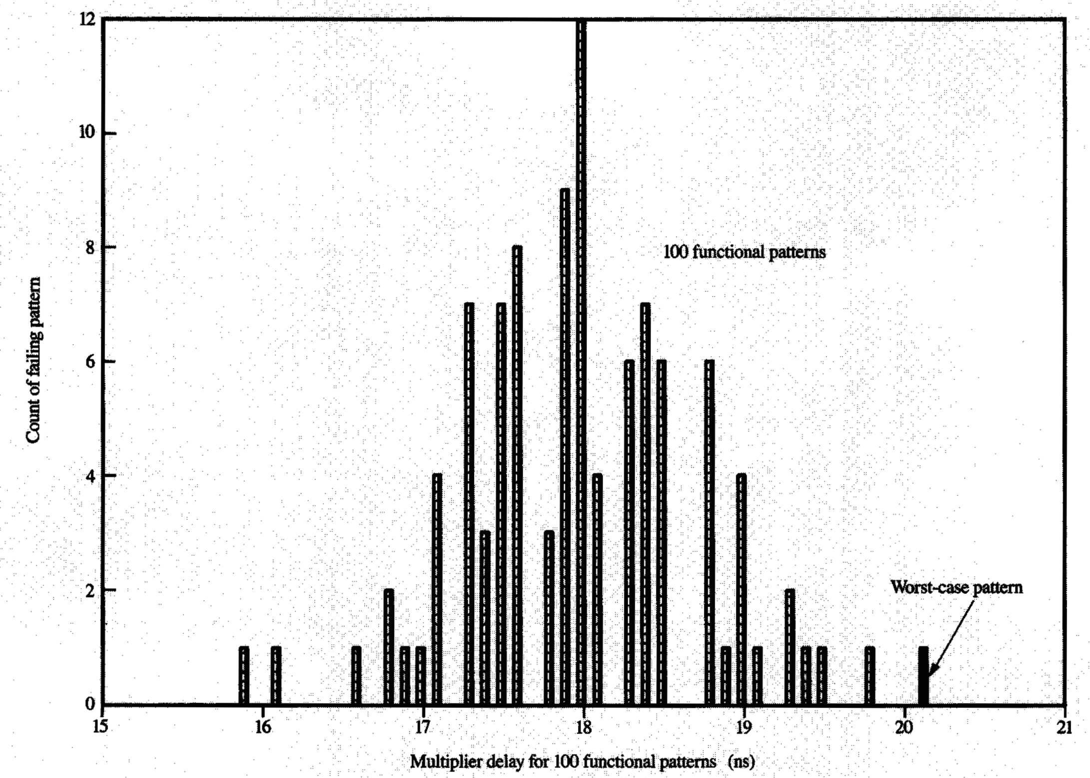


Figure 10

Pattern-dependent delay.

adder circuit must be optimized. Two different adder cells are used within the multiplier array. This scheme eliminates a delay of one inverter in each array row. **Figures 8(a)** and **8(b)** show the two full adders with alternating polarities for the carry-in (CI) and the input signal A. Input B is fed by the sum output of a previous adder and has the same polarity throughout. The circuits are nearly identical; only the two connections X and X-not are different. This eases the integration of the full adders into an array. A full adder cell consists of three inverters and twelve transmission gates. Eight transmission gates are used to generate two exclusive-OR functions, while four gates generate the carry-out function. If one follows the delay paths from one full adder to the next, a maximum of three transmission gates in series between two inverters are encountered.

Worst-case paths are, however, caused by irregularities due to the necessary AND-function between the full adders at the right edge (Figure 6). The circuit for this block is shown in **Figure 9**. The transmission gates T2, T3, T10, T11, T18, T19, T22, and T23 perform the AND-function, which is integrated into the first exclusive-OR function of the full adder. This integration makes the additional delay modest.

Test results

The typical delay of the multiplier macrocell from input to output register was measured to be 18 ns. This delay is the sum of two multiplexer delays and 14 carry-save adder delays inside the arrays, and of two carry-save adder delays and an on-chip driver delay outside the arrays. It also includes the setup time for the registers. The pattern dependency of the multiplier delay can be seen in **Figure 10**. The delays, which vary between 16 ns and 19.5 ns, are the result of applying 100 functional patterns. Finally, **Figure 11** shows a distribution of multiplier delays resulting from the application of a worst-case pattern to several chips out of five experimental lots.

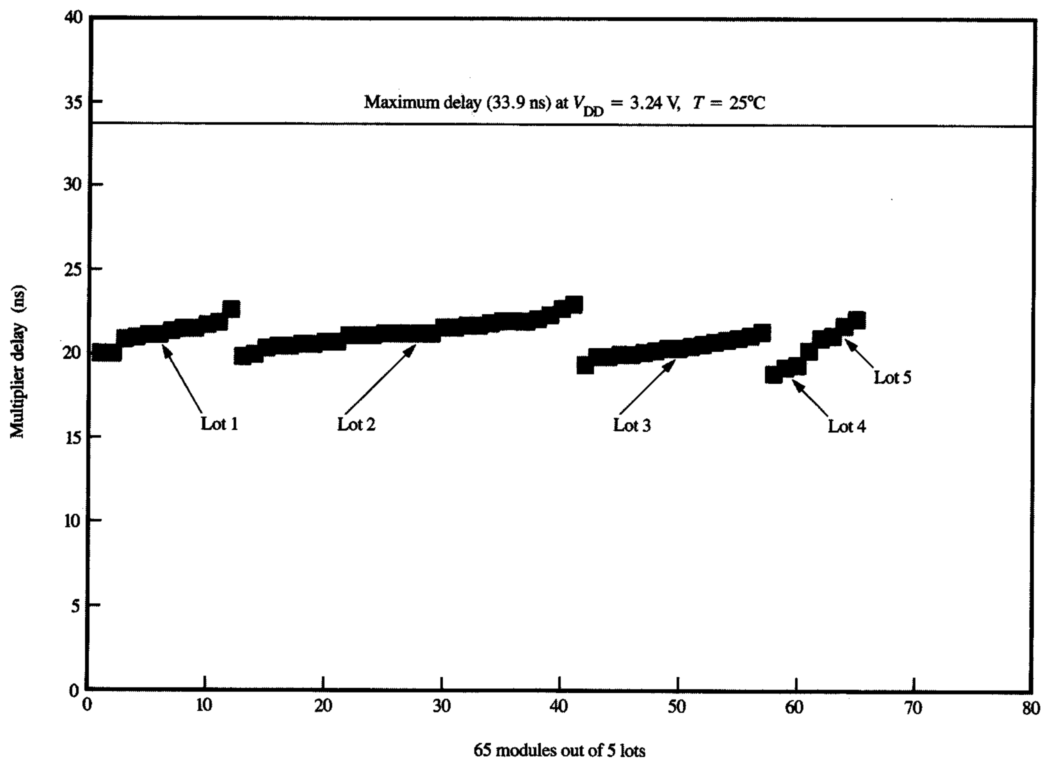


Figure 11

Worst-case delay distribution.

Verification and simulation

For simulation and verification purposes, a one-to-one logic gate model of the multiplier was written and then was merged into the entire floating-point model for circuit simulation. With this multiplier model, data were generated to further test and stress the multiplier macro, which was built on a test site. Because the macro was designed manually, additional effort was needed to check the correctness of the design. To this end, a topology-checking program named MACH1 from our Engineering Design System (EDS) was used to check the transistor shapes against the logic gate model. A library of transistor shapes was created for defining the correspondence to logic gates. The program then checked the shape connections for correct function. By using both simulation and shape checking, a correct multiplier macro was built and merged onto the floating-point chip.

Binary multiplication

Although not a floating-point instruction, binary integer multiplication is also implemented on the floating-point processor to improve performance. The floating-point processor is very highly coupled to the CPU and is an integral part of it. Because the Booth encoders are implemented outside the multiplier macro, modification to the Booth coefficients can be easily made to match the requirements for signed multiplication [5].

- *Division*

Division is performed by the floating-point processor using a modified Newton convergence method. The starting seed for the division comes from a table which is seven bits wide. In the following, we describe only operations on the fractional part of the operands. Exponent calculation is simple and is calculated separately.

Let A and B be the fractional parts of the dividend and divisor, respectively. Then let A and B be bit-normalized; i.e., the most significant bit is nonzero (for hex-format, a bit-normalization must be done before processing):

$$A = 0.1a_1a_2a_3 \cdots a_{55},$$
$$B = 0.1b_1b_2b_3 \cdots b_{55}. \quad (1)$$

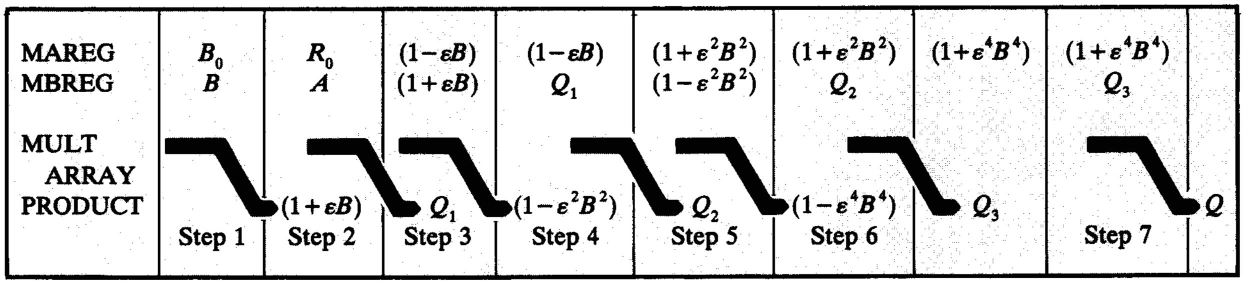


Figure 12

Example divide in pipeline structure.

Step 1 $$B \cdot R_0 = 0.73216312 \times 1.36$$
$$= 0.99574184 = 1 + \varepsilon B,$$
$$1 - \varepsilon B = 1.00425816.$$

Step 2 $$A \cdot R_0 = 0.59786552 \times 1.36$$
$$= 0.81309710.$$

Step 3 $$(1 + \varepsilon B)(1 - \varepsilon B) = 0.99998186 = 1 - \varepsilon^2 B^2,$$
$$1 + \varepsilon^2 B^2 = 1.00001814.$$

Step 4 $$A \cdot R_0 \cdot (1 - \varepsilon B) = 0.81309710 \times 1.00425816$$
$$= 0.81655939.$$

Step 5 $$(1 - \varepsilon^2 B^2)(1 + \varepsilon^2 B^2) = 0.99998186 \times 1.00001814$$
$$= 0.99999999 = 1 - \varepsilon^4 B^4,$$
$$1 + \varepsilon^4 B^4 = 1.00000001.$$

Step 6 $$A \cdot R_0 \cdot (1 - \varepsilon B)(1 + \varepsilon^2 B^2) = 0.81655939 \times 1.00001814$$
$$= 0.81657420.$$

Step 7 $$A \cdot R_0 \cdot (1 - \varepsilon B)(1 + \varepsilon^2 B^2)(1 + \varepsilon^4 B^4) = 0.81657420 \times 1.00000001$$
$$= 0.81657420 = Q.$$

Comparing multiplications:

$B \cdot Q = 0.73216312 \times 0.81657420 = 0.59786551 < A,$
$B(Q + 1LSB) = 0.73216312 \times 0.81657421 = 0.59786552 = A.$

However, the low part of the product, not shown here, is not zero, so that the full product is greater than A.

Thus, the correct result is

$Q = 0.81657420.$

Note: In hex format, the result must be corrected according to the bit normalization done before processing. This step may increment the result exponent by one in the low-order bit.

Figure 13

Example divide A/B for $A = 0.59786552$ and $B = 0.73216312$.

Let R_0 be the start seed, which is the seven-bit approximation of the inverse of B:

$$R_0 = \frac{1}{B} + \varepsilon; \tag{2}$$

ε has at least seven leading zero bits.

Division is performed as follows:

$$\frac{A}{B} = \frac{A \cdot R_0}{B \cdot R_0} = \frac{A \cdot R_0}{B\left(\frac{1}{B} + \varepsilon\right)} = \frac{A \cdot R_0}{1 + \varepsilon B}$$

$$= \frac{A \cdot R_0(1 - \varepsilon B)}{(1 + \varepsilon B)(1 - \varepsilon B)} = \frac{A \cdot R_0(1 - \varepsilon B)}{1 - \varepsilon^2 B^2}$$

$$= \frac{A \cdot R_0(1 - \varepsilon B)(1 + \varepsilon^2 B^2)}{(1 - \varepsilon^2 B^2)(1 + \varepsilon^2 B^2)} = \frac{A \cdot R_0(1 - \varepsilon B)(1 + \varepsilon^2 B^2)}{1 - \varepsilon^4 B^4}$$

$$= \frac{A \cdot R_0(1 - \varepsilon B)(1 + \varepsilon^2 B^2)(1 + \varepsilon^4 B^4)}{(1 - \varepsilon^4 B^4)(1 + \varepsilon^4 B^4)}$$

$$= \frac{A \cdot R_0(1 - \varepsilon B)(1 + \varepsilon^2 B^2)(1 + \varepsilon^4 B^4)}{1 - \varepsilon^8 B^8}. \tag{3}$$

Because $\varepsilon^8 B^8$ has at least 64 leading zero bits, the quotient of A/B can be approximated by

$$\frac{A}{B} \approx A \cdot R_0(1 - \varepsilon B)(1 + \varepsilon^2 B^2)(1 + \varepsilon^4 B^4) = Q. \tag{4}$$

We need to perform only seven multiplications because the last multiplication result, which is $(1 - \varepsilon^8 B^8)$, can be discarded. It was mathematically proven that the approximation so found is at most one least significant bit away from the correct result. The precise result of the division is then determined by making the two comparing multiplications

$$A \gtrless B \cdot Q$$

and

$$A \gtrless B(Q + 1LSB), \tag{5}$$

where LSB indicates the least significant bit.

The terms $(1 - \varepsilon B)$, $(1 + \varepsilon^2 B^2)$, and $(1 + \varepsilon^4 B^4)$ used in Equation (3) are derived from the terms $(1 + \varepsilon B)$, $(1 - \varepsilon^2 B^2)$, and $(1 - \varepsilon^4 B^4)$ respectively, using special built-in circuitry. Hardware needed is minimal, whereas for the same function others need a table-lookup ROM [6].

The sequence of the division is shown in **Figure 12.** The complete division including the correcting steps is implemented in hardware and requires 14 cycles. An example of a divide using decimal numbers is illustrated in **Figure 13**. Let $A = 0.59786552$ and $B = 0.73216312$. The divide table delivers the rounded inverse of the truncated B operand, which is 0.73. Thus, $R_0 = 1/0.73 = 1.36$.

Table 2 Performance summary—floating-point chip.

Clock	37 MHz maximum
Register-to-register instruction	1 cycle typical
Register-to-memory instruction	3 cycles typical
LINPACK [7] performance	4.1 MFLOPS
Power dissipation	2 W

Summary

Figure 14 shows a wiring plot of the chip. Wiring amounted to more than 30 meters at the first iteration, resulting in an unwirable design. After two iterations, including new floor-planning, a design requiring only 18 meters of wire was produced. The chip measures 12.7 mm × 12.7 mm and contains more than 70 000 wired cells (equivalent inverter). All single-, double-, and extended-precision floating-point instructions, and the binary integer multiplication, are performed as defined by the IBM System/390 principles of operation. The data types are short format (one sign bit, seven exponent bits, and 24 fraction bits), long format (one sign bit, seven exponent bits, and 56 fraction bits), and extended format (one sign bit, seven exponent bits, and 112 fraction bits). Instructions that are heavily used, such as addition, multiplication, load, and store, are performed in one cycle, except for extended operands. In pipelining, mode, long results can be sdelivered every cycle, which is 27 ns. **Table 2** shows a performance summary of the floating-point chip.

References

1. H. Schettler, K. Getzlaff, K. Klein, C. W. Starke, J. Wilczynski, and A. Bhattacharyya, "A CMOS Mainframe Processor with a 0.5 μm Channel Length," *IEEE J. Solid State Circuits* **25**, 1166 (1990).
2. O. L. McSorley, "High Speed Arithmetic in Binary Computers," *Proc. IRE* **49**, 67 (1961).
3. C. S. Wallace, "A Suggestion for Parallel Multiplier," *IEEE Trans. Electron. Computers* **EC-13**, 14 (1964).
4. K. G. Getzlaff, S. Dao-Trong, and K. Helwig, "Multiplizierwerk (Multiplier)," European Patent Office, Reg. Number 89102956.3, February 1989.
5. A. D. Booth, "A Signed Binary Multiplication Algorithm," *Quart. J. Mech. Appl. Math.* **IV**, Part 2, 163 (1951).
6. K. Kaneko, T. Okamoto, M. Nakajima, Y. Nakakura, S. Gokita, J. Nishikawa, Y. Tanikawa, and H. Kadota, "A VLSI RISC with 20 MFLOPS Peak, 64-Bit Floating Point Unit," *IEEE J. Solid State Circuits* **24**, 1331 (1989).
7. J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

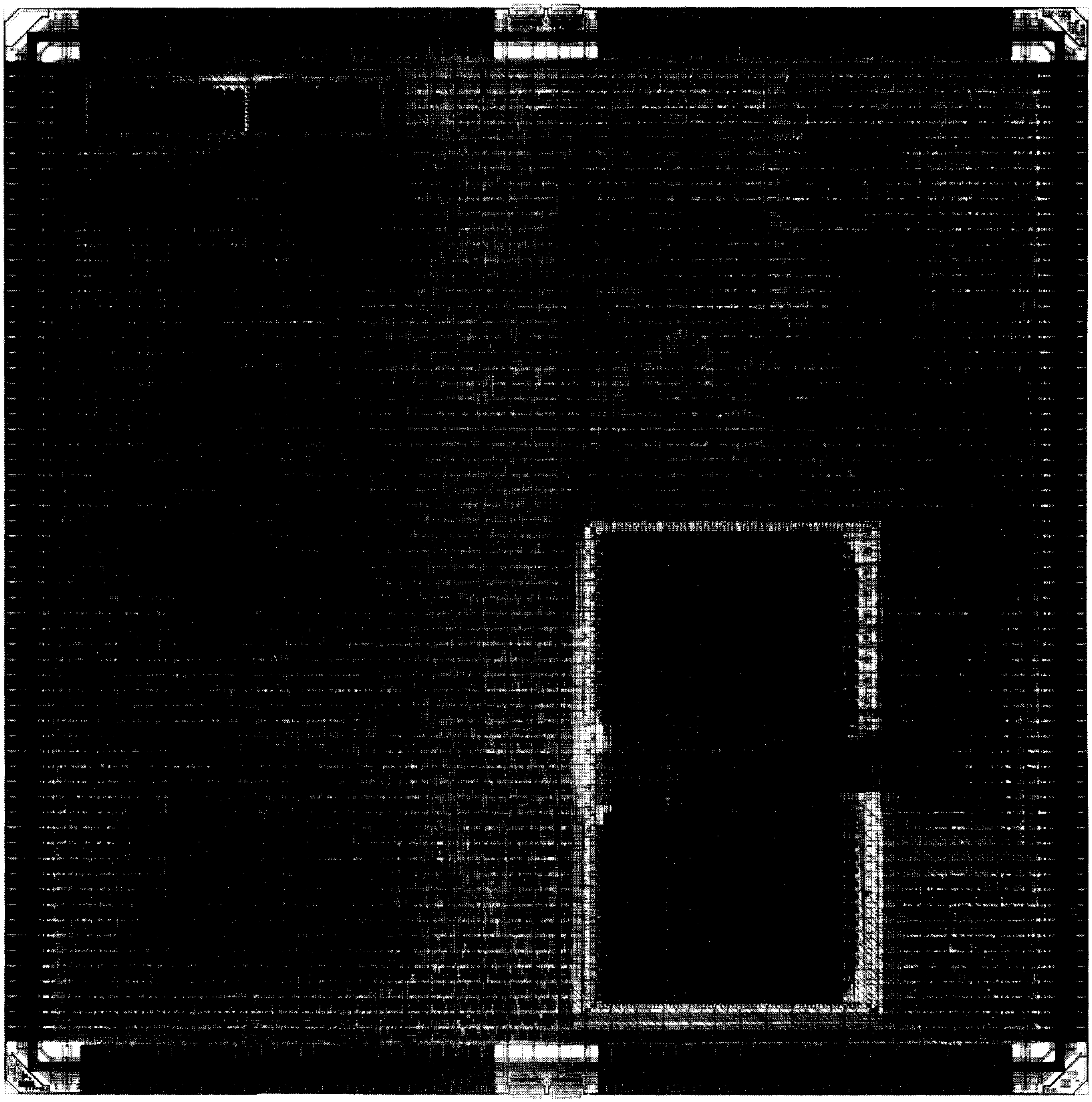


Figure 14

Floating-point chip wiring.

Received April 18, 1991; accepted for publication September 18, 1991

Son Dao-Trong *IBM Germany, Schoenaicherstrasse 220, 7030 Boeblingen, Germany (DAOTRONG at BOEVM2).* Dr. Dao-Trong received his Dipl.-Ing. degree in electrical engineering from the University of Karlsruhe. In 1979 he joined the University, where he worked on logic synthesis tools in a gate-array environment. He received his Doctor-Ing. degree in 1984 and joined the IBM Laboratory in Boeblingen, where he first worked on packaging design and noise analysis. Since 1987 he has worked on floating-point design as a team leader. In 1989, Dr. Dao-Trong received a patent on the multiplier array which is embedded in the floating-point chip of the recent Enterprise System/9000 9221 models. He is currently working on the design of the next model, stressing design methodology.

Klaus Helwig *IBM Germany, Schoenaicherstrasse 220, 7030 Boeblingen, Germany (HELWIG at BOEVM4).* Mr. Helwig received the Dipl.-Ing. degree in electrical engineering from the Technical University of Vienna, Austria, in 1965. From 1966 to 1968, he was employed with Siemens AG, Munich, Germany, where he was engaged in the development of MOS transistors. In 1968, he joined the Components Group of the IBM Laboratories, Boeblingen, Germany, working on digital integrated circuit designs especially for monolithic memory applications. During the past years, Mr. Helwig has been engaged in the design of CMOS memory and logic arrays.

6/5/3 #4 AB
Resp/.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Gioquindo et al.

Confirmation No.: 3569

Serial No.: 09/977,799

Group Art Unit: 2141

Filed: 10/15/01

Examiner: Jaroenchonwanit, Bunjob

Title: COMMUNICATION BETWEEN MULTIPLE PARTITIONS EMPLOYING HOST-NETWORK INTERFACE

Certificate of Mailing

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Mail Stop Non-Fee Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on May 23, 2003.

Kevin P. Radigan
Kevin P. Radigan, Esq.
Attorney for Applicants
Reg. No. 31,789

Date of Signature: May 23, 2003

Mail Stop Non-Fee Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RECEIVED
MAY 2 9 2003
Technology Center 2100

Response to Office Action

Dear Sir:

This paper is submitted in reply to the Office Action mailed March 3, 2003, in connection with the above-designated application. A Response to the Office Action is initially due on or before June 3, 2003, and thus, this Response is timely filed. Reconsideration and allowance of all claims are respectfully requested.

Remarks

Applicants gratefully acknowledge the indication of allowability of claims 2-4, 7-9 and 12-14 if rewritten in independent form including all the limitations of the base claim and any intervening claims. Since applicants believe that the remaining claims are in condition for allowance over the applied art, applicants are presently deferring rewriting these dependent claims in independent form. Claims 1-15 remain pending.

Claims, 1, 5-6, 10-11 and 15 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Ratcliff et al. (U.S. Patent No. 5,740,438). This rejection is respectfully traversed and reconsideration thereof is requested.

In accordance with the pending independent claims (i.e., claims 1, 6 & 11), applicants disclose a communications technique implemented within a host-network interface for use in a mainframe class data processing system having multiple partitions and a port connected to a network. The communications technique includes saving at the host-network interface an internet protocol (IP) address of at least one of the multiple partitions of the mainframe class data processing system. An IP datagram is generated at a first partition of the multiple partitions to be forwarded to a second partition of the multiple partitions of the mainframe class data processing system. This forwarding includes using a destination IP address. Applicants' technique further includes determining whether the destination IP address for the IP datagram comprises an IP address saved at the host-network interface for the at least one partition, and if so, forwarding the IP datagram directly from the first partition to the second partition of the multiple partitions without employing the network. Effectively, applicants' technique intercepts the IP datagram from the first partition, uses the associated destination IP address for the IP datagram to establish that the IP datagram is to be forwarded to a second partition of the mainframe class data processing system, and then forwards the IP datagram from the first partition to the second partition <u>without employing the network</u>.

Before discussing the applied art relative to applicants' above-summarized invention, it is respectfully submitted that an "obviousness" determination requires an evaluation of whether the

prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art. In evaluating claimed subject matter as a whole, the Federal Circuit has expressly mandated that functional claim language be considered in evaluating a claim relative to the prior art. Applicants respectfully submit that the application of these standards to the independent claims presented herewith leads to the conclusion that the recited subject matter would not have been obvious to one of ordinary skill in the art based on the applied patent.

For example, applicants' independent claims each recite a technique wherein an IP datagram from a first partition of multiple partitions within a mainframe class data processing system is forwarded directly from the first partition to a second partition of the mainframe class data processing system using the associated destination IP address and <u>without employing the network</u>. As known in the art, a destination internet protocol (IP) address accompanies data forwarded across the network and designates the particular address to which the data is to be sent. Thus, the destination IP address essentially comprises a network address or transport layer address designating the destination to which the data is to be sent. Applicants' technique includes saving at the host-network interface, which is coupled between the mainframe class data processing system having the multiple partitions and a port to a network, an internet protocol address of at least one of the partitions in the mainframe class data processing system. The associated destination IP address with the IP datagram is then compared to the saved IP address(es) at the host-network interface, and if a match is found, the IP datagram is forwarded directly from the first partition to the second partition of the mainframe class data processing system without ever going out onto the network (i.e., "without employing the network"). Applicants respectfully submit that no such functionality is taught, suggested or implied by the Ratcliff et al. patent.

The Ratcliff et al. patent, of which there is partially overlapping inventorship with the present application, describes techniques for network communications involving multiple partitions. In particular, a table is established in a mainframe class data processing system having multiple partitions and a port to a network. The table defines communication paths between the port to the network and the at least two partitions of the processing system. Each

partition has at least one application executing therein and the communications paths are defined thereto. Data frames are passed between the network and the applications within the partitions and through the port to the network and along the communications paths defined in the table such that network communications are effected.

Applicants respectfully submit that a careful reading of the Ratcliff et al. patent fails to uncover any suggestion or implication of their technique as recited in the independent claims presented herewith. For example, there is no suggestion or implication of functionality in the Ratcliff et al. patent for determining whether a destination IP address for an IP datagram comprises an IP address saved at a host-network interface for at least one partition, and if so, for forwarding the IP datagram directly from the first partition to the second partition without employing the network. Ratcliff et al. describes transmitting data frames out onto a network, and forwarding data frames from a network to an appropriate destination address within the mainframe class data processing system. The Ratcliff et al. patent does not suggest or imply any technique, other than the conventional approach, for forwarding an IP datagram from a first partition to a second partition of a mainframe class data processing system without employing the network.

In stating the obviousness rejection, the Office Action references at page 3, line 6, column 10, lines 5-18 of Ratcliff et al., which are reproduced below for the Examiner's convenience:

> (a) receiving an inbound data frame from said network through said port to said network, said inbound data frame having a logical destination address;
>
> (b) extracting said logical destination address from said inbound data frame and looking up in said table a destination application of the at least one application executing within each partition of the at least two partitions having a corresponding logical address to said logical destination address; and
>
> (c) if said destination application having said corresponding logical address is found in said table, passing said inbound data frame to said destination application such that said network communications is effected between said network and said destination application.

The above-cited lines of Ratcliff et al. set forth a method of network communications implemented within a host-network interface for use in a mainframe class data processing system having multiple partitions. This method includes receiving an inbound data frame from the network. In contrast, applicants' invention recites generating an IP datagram at a first partition of the multiple partitions of the mainframe class data processing system to be forwarded to a second partition of the multiple partitions using a destination IP address. Applicants' data frame is thus generated within the multiple partitions of the mainframe class data processing system and is not received from across the network as in Ratcliff et al. Ratcliff et al. concludes claim 8, at lines 16-18 by reciting "... such that said network communications is effected between said network and said destination application." This teaching of Ratcliff et al. is clearly different from applicants' recited functionality, wherein applicants determine whether the destination IP address for the IP datagram comprises an IP address saved at the host-network interface for the at least one partition, and if so, forward the IP datagram directly from the first partition to the second partition of the multiple partitions without employing the network.

Applicants expressly teach and recite functionality for determining whether the destination IP address for the IP datagram comprises an IP address saved at the host-network interface for the at least one partition of the mainframe class data processing system, and if so, for forwarding the IP datagram directly from the first partition to the second partition without employing the network. The intelligence to accomplish this forwarding is not taught, suggested or implied by Ratcliff et al. The cited language at column 10 expressly discusses network communications being effectuated between the network and the destination application, and includes receiving an inbound data frame from the network, which is clearly distinct from the communications technique recited in applicants' independent claims. In view of this, and since there is no justification in the art for modifying the Ratcliff et al. teachings, applicants respectfully submit that the independent claims presented would not have been obvious to one of ordinary skill in the art based thereon.

The dependent claims still at issue are believed allowable for the same reasons as their respective independent claims, as well as for their own additional characterizations. With respect to the citation in the specification of various patents and publications, applicants respectfully submit that this material is cited, and in part incorporated, for background information only. The art cited in applicants' Information Disclosure Citation filed with the application is believed by applicants to comprise the most relevant art.

For all the above reasons, applicants respectfully request reconsideration and allowance of all claims presented herewith.

Applicants' undersigned attorney is available should the Examiner wish to discuss this application further.

Respectfully submitted,

Kevin P. Radigan

Kevin P. Radigan, Esq.
Attorney for Applicants
Registration No. 31,789

Dated: May 23, 2003

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

24

AMENDMENT TRANSMITTAL LETTER (Large Entity) Applicant(s): **Gioquindo et al.**		Docket No. **POU919980100US2**
Serial No.	Filing Date **09/977,799**	Examiner **Jaroenchonwanit, Bunjob**	Group Art Unit **2141**

6/3/3

OIPE MAY 2 8 2003 PATENT & TRADEMARK OFFICE

Invention: **COMMUNICATION BETWEEN MULTIPLE PARTITIONS EMPLOYING HOST-NETWORK INTERFACE**

RECEIVED
MAY 2 9 2003
Technology Center 2100

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

Transmitted herewith is an amendment in the above-identified application.

The fee has been calculated and is transmitted as shown below.

CLAIMS AS AMENDED

	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST # PREV. PAID FOR		NUMBER EXTRA CLAIMS PRESENT		RATE	ADDITIONAL FEE
TOTAL CLAIMS	20	-	20	=	0	x	$18.00	$0.00
INDEP. CLAIMS	3	-	3	=	0	x	$84.00	$0.00
Multiple Dependent Claims (check if applicable) ☐								$0.00
TOTAL ADDITIONAL FEE FOR THIS AMENDMENT								$0.00

☒ No additional fee is required for amendment.

☐ Please charge Deposit Account No. in the amount of
A duplicate copy of this sheet is enclosed.

☐ A check in the amount of to cover the filing fee is enclosed.

☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. **09-0457 (IBM)**
A duplicate copy of this sheet is enclosed.

- ☒ Any additional filing fees required under 37 C.F.R. 1.16.
- ☒ Any patent application processing fees under 37 CFR 1.17.

Kevin P. Radigan
Signature

Dated: **May 23, 2003**

Kevin P. Radigan, Esq.
Registration No. 31,789
Heslin Rothenberg Farley & Mesiti P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579

I certify that this document and fee is being deposited on 5/23/03 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Kevin P. Radigan
Signature of Person Mailing Correspondence

Kevin P. Radigan
Typed or Printed Name of Person Mailing Correspondence

cc:

Physical media, layer 1

The physical network (also called the *physical layer*) begins at the network interface card (NIC). The NIC is effectively a method of connecting the internal data bus of a computer to the external media (cables) of the network.

In the case of z/OS there is essentially only one NIC: the Open Systems Adapter (OSA) card. The OSA card is discussed in detail in the topic on hardware connectivity. There are other ways of attaching a network to a z/OS host, but they are seldom used. These other interfaces (parallel channel, ESCON, and Coupling Facility attachments) are discussed in the topic on hardware connectivity.

Network interface card (NIC)
Although the OSA card is the only NIC for z/OS, this is a bit of an understatement. The OSA card variants support Ethernet in all of its current implementations. This means that it physically can connect to either a fiber optic cable or a copper (twisted pair) media. When connected to the latter, the ubiquitous RJ-45 is the connection type used.

Parent topic: Network layers and protocols review

Related information
Hardware connectivity on the mainframe

URL: http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.znetwork.doc/znetwork_24.html

Hardware connectivity on the mainframe

Network connections can be made in several different fashions. The mainframe originally relied upon the channel subsystem to offload I/O processing to channel programs. DASD is still accessed using ESCON channels, but for networking connectivity, OSA-Express cards offer better performance and availability.

The OSA-Express and OSA-Express2 cards provide redundancy capability, as well as throughput improvements when running in QDIO mode. QDIO mode allows direct access to central memory. QDIO mode can be emulated within a CPC by allowing memory to memory data transfer among LPARs running z/VM, Linux, or z/OS.

Connections for the mainframe

The design intention of the mainframe, and most of its evolution, is for the mainframe to be a highly available transaction processing server. Obviously, central processing capabilities are evolving to handle more and more transactions. However, in order to be an effective transaction processing server, there must be a proportional capability of moving data in and out of the central processor complex rapidly (CPC, the physical collection of hardware that consists of main storage, one or more central processors, timers, and channels). The result is that the I/O (input/output) options, capabilities and configuration choices of an IBM mainframe are varied, complex, and very performance oriented.

Mainframe computers are probably unique in that they require a Hardware Management Console, or HMC. The HMC is a separate interface to the central processor complex that is used for hardware configuration operations. It also provides an interface to the z/OS system console.

Channel subsystem (CSS)
The heart of moving data into and out of a mainframe host is the *channel subsystem*, or CSS. The CSS is, from a central processor standpoint, independent of the processors of the mainframe host itself. This means that input/output (I/O) within a mainframe host can be done asynchronously.

The mainframe channel subsystem and network links
A CHPID no longer directly corresponds to a hardware channel, and CHPID numbers may be arbitrarily assigned. A hardware channel is now identified by a physical channel identifier, or PCHID.

Hardware channels
There are effectively three ways that network traffic can travel between an external network and a z/OS host: through a channel-command word channel, a coupling channel, or a QDIO channel.

HiperSockets
Mainframe HiperSockets is a technology that provides high-speed TCP/IP connectivity within a central processor complex. It eliminates the need for any physical cabling or external

networking connection between servers running in different LPARs.

The I/O cage
The connections to the central processor complex are made in a physical area of the processor frame called an *I/O cage*. Within the cage, OSA cards and memory modules (and other devices) are physically attached to the central processor complex. Parallel, FICON and ESCON connections are all made within the cage as well, using an adapter card.

Parent topic: Introduction to networking on the mainframe

URL: http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.znetwork.doc/znetwork_57.html

US005740438A

United States Patent [19]

Ratcliff et al.

[11] Patent Number: 5,740,438

[45] Date of Patent: Apr. 14, 1998

[54] METHODS AND SYSTEM FOR NETWORK COMMUNICATIONS OF MULTIPLE PARTITIONS

[75] Inventors: **Bruce Henry Ratcliff; Anthony Robert Sager**, both of Red Hook; **Stephen Roger Valley**, Valatie, all of N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **414,851**

[22] Filed: **Mar. 31, 1995**

[51] **Int. Cl.**[6] **G06F 13/14**

[52] **U.S. Cl.** **395/680**; 370/463

[58] **Field of Search** 395/650, 416, 395/417, 418, 680; 370/463, 437

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,768,150	8/1988	Chang et al.	395/682
4,780,821	10/1988	Crossley	364/200
4,835,674	5/1989	Collins et al.	364/200
4,949,248	8/1990	Caro	364/200
4,985,892	1/1991	Camarata	370/123
5,265,239	11/1993	Ardolino	395/500
5,299,193	3/1994	Szczepanek	370/463
5,313,592	5/1994	Buondonno et al.	395/284
5,315,711	5/1994	Barone et al.	395/275
5,365,606	11/1994	Brocker et al.	395/650
5,448,566	9/1995	Richter et al.	370/431
5,471,474	11/1995	Grobicki et al.	370/437
5,535,338	7/1996	Krause et al.	395/200.2
5,546,584	8/1996	Lundin et al.	395/683

OTHER PUBLICATIONS

Accetta et al., "Mach: A New Kernel Foundation for UNIX Development", Proceedings of the Summer 1986 USENIX Conference, Atlanta, GA, pp. 93–112, 1986.

Black et al., "Microkernel Operating System Architecture and Mach", Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures, Seattle Washington, pp. 11–30, Apr. 1992.

Bricker et al., "Architectural issues in microkernel-based operating systems: the CHORUS experience", Computer Communications, v. 14, No. 6, pp. 347–357, Aug. 1991.

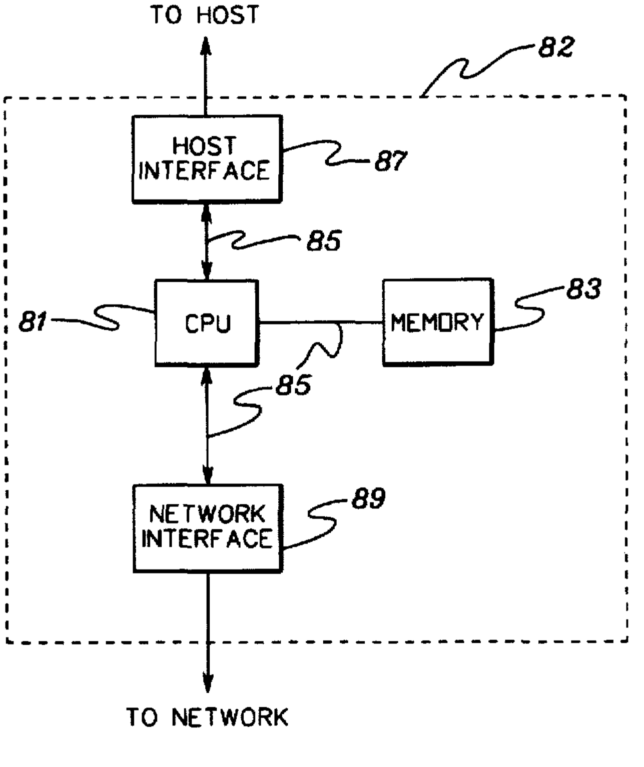
Proceedings of the USENIX Workshop on Micro-Kernels & Other Kernel Architectures, Apr. 1992.

Primary Examiner—Tod R. Swann
Assistant Examiner—Christopher S. Chow
Attorney, Agent, or Firm—Lawrence D. Cutter; Heslin & Rothenberg, PC

[57] **ABSTRACT**

In a mainframe class data processing system having multiple partitions and a port to a network, a table is established. The table defines communications paths between the port to the network and at least two partitions of the multiple partitions. More specifically, each partition has at least one application executing therewithin and the communications paths are defined thereto. Data frames are passed between the network and the applications within the partitions through the port to the network and along the communications paths defined in the table such that network communications is effected.

22 Claims, 6 Drawing Sheets



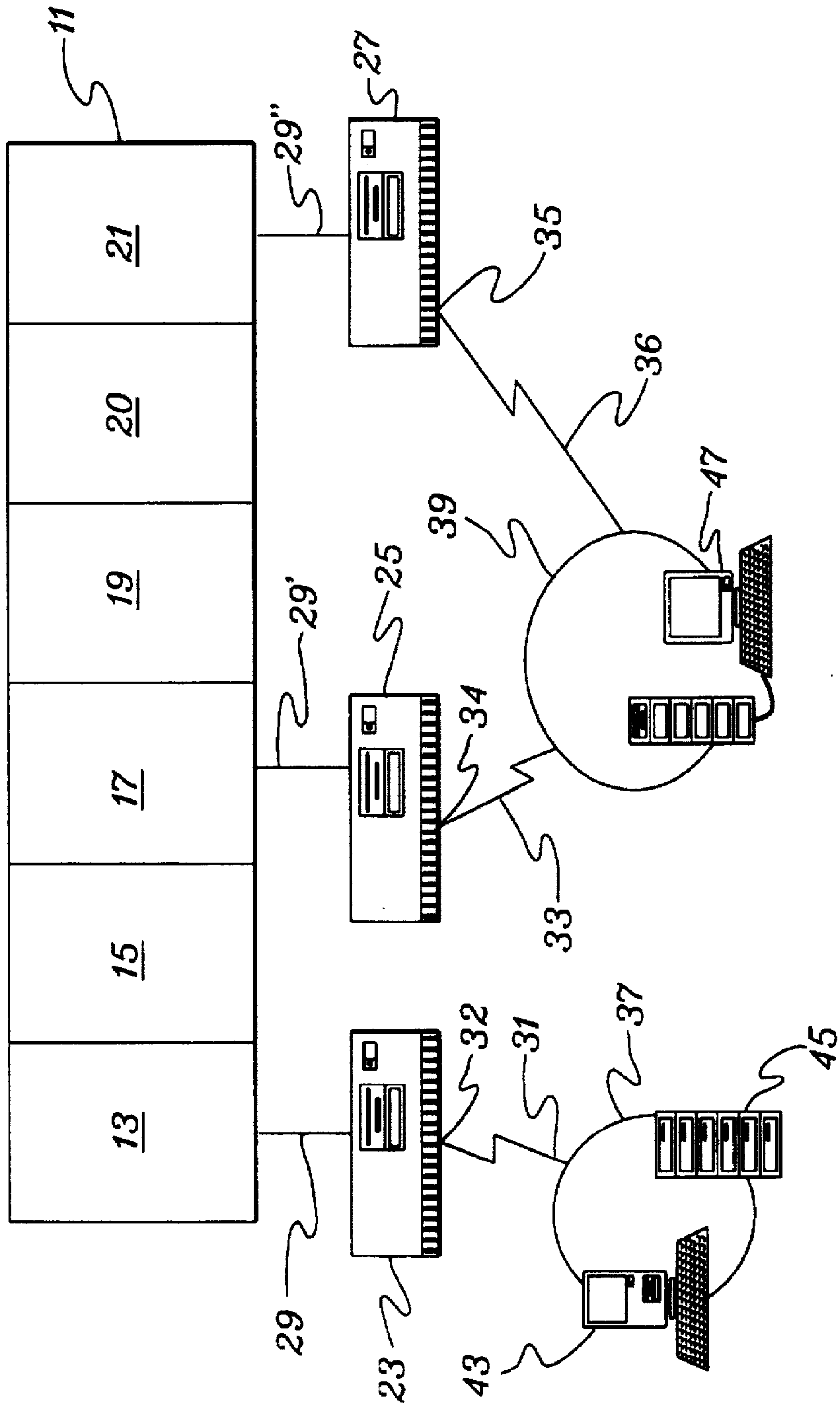


fig. 1
(PRIOR ART)

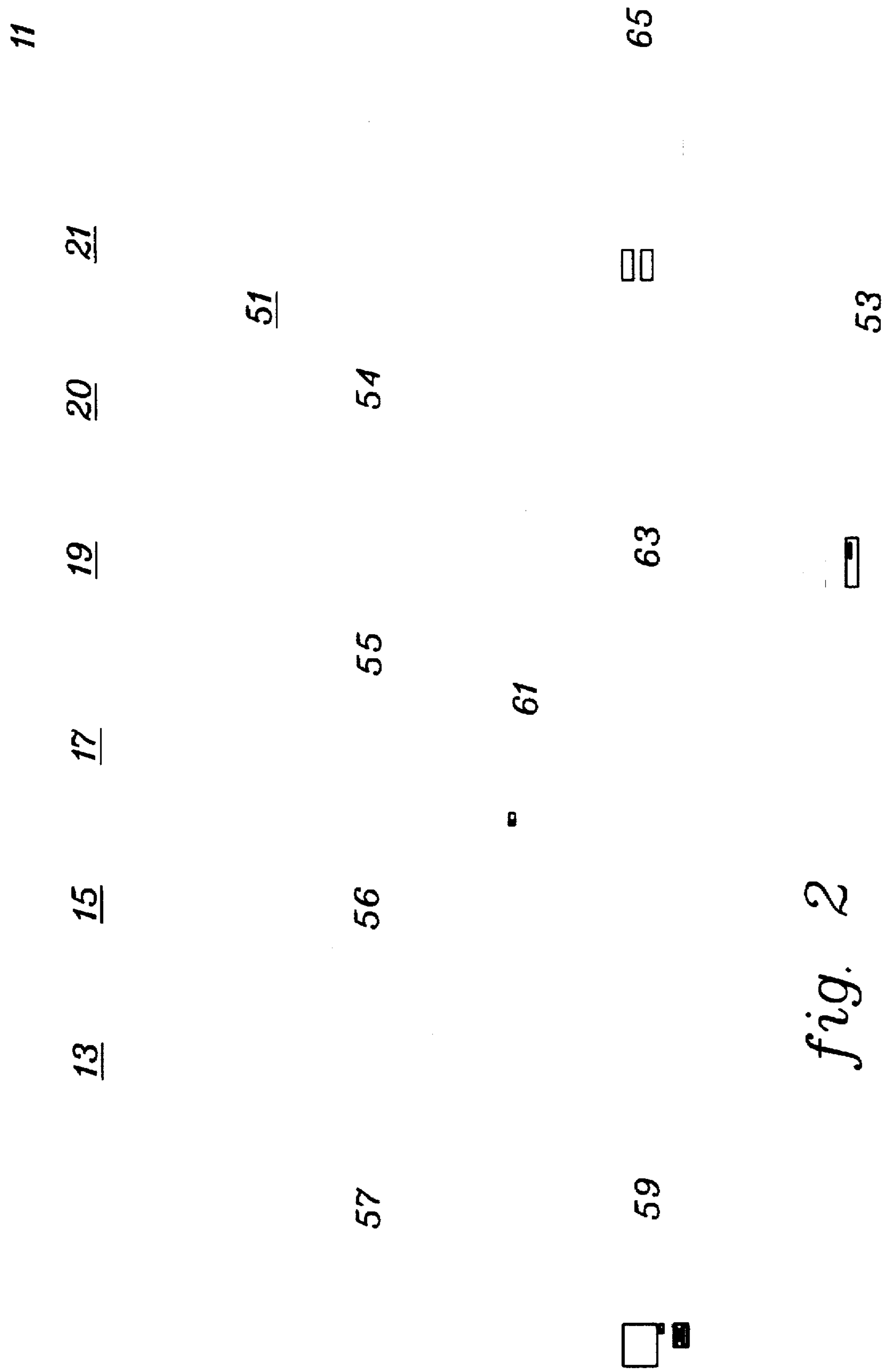


fig. 2

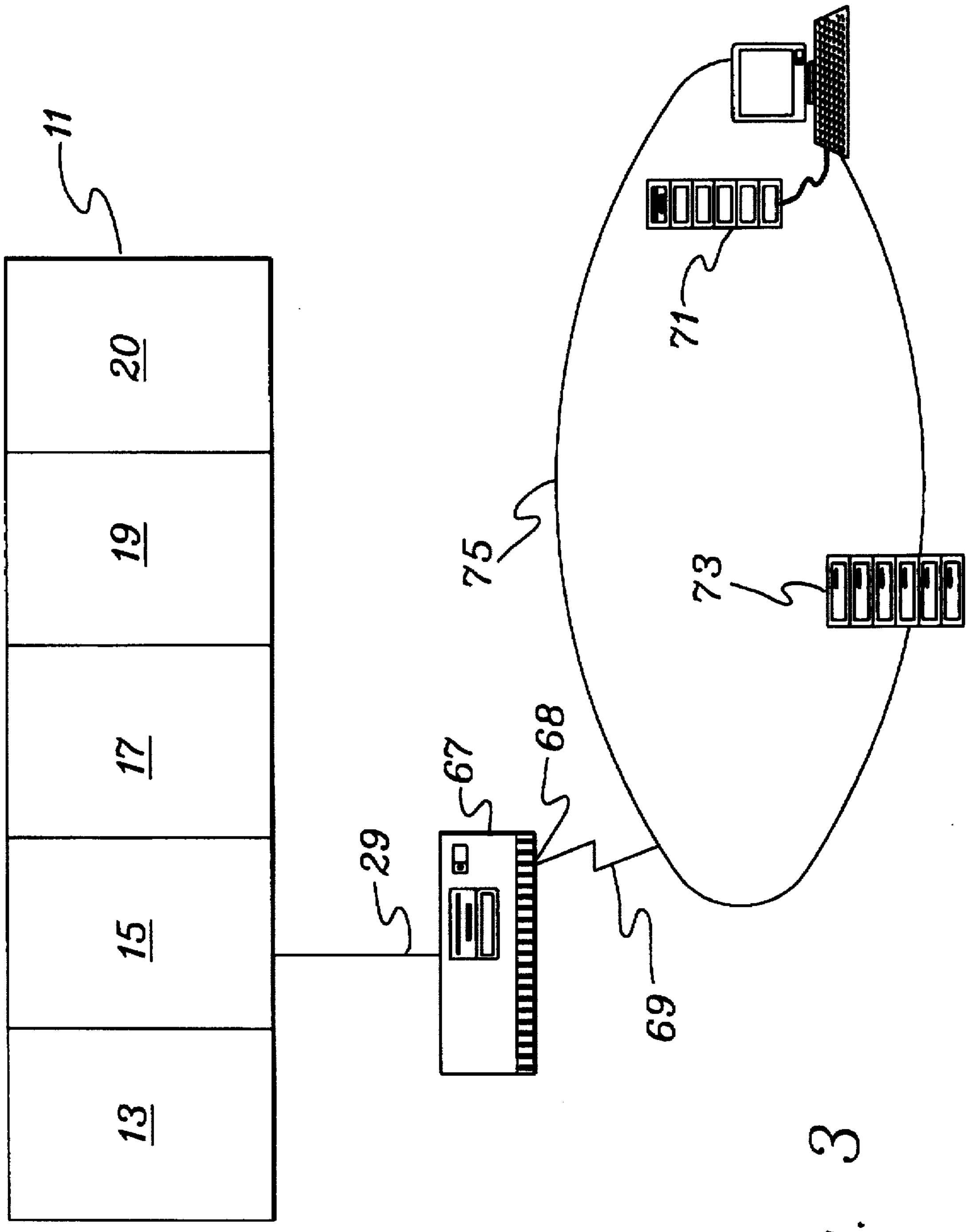


fig. 3

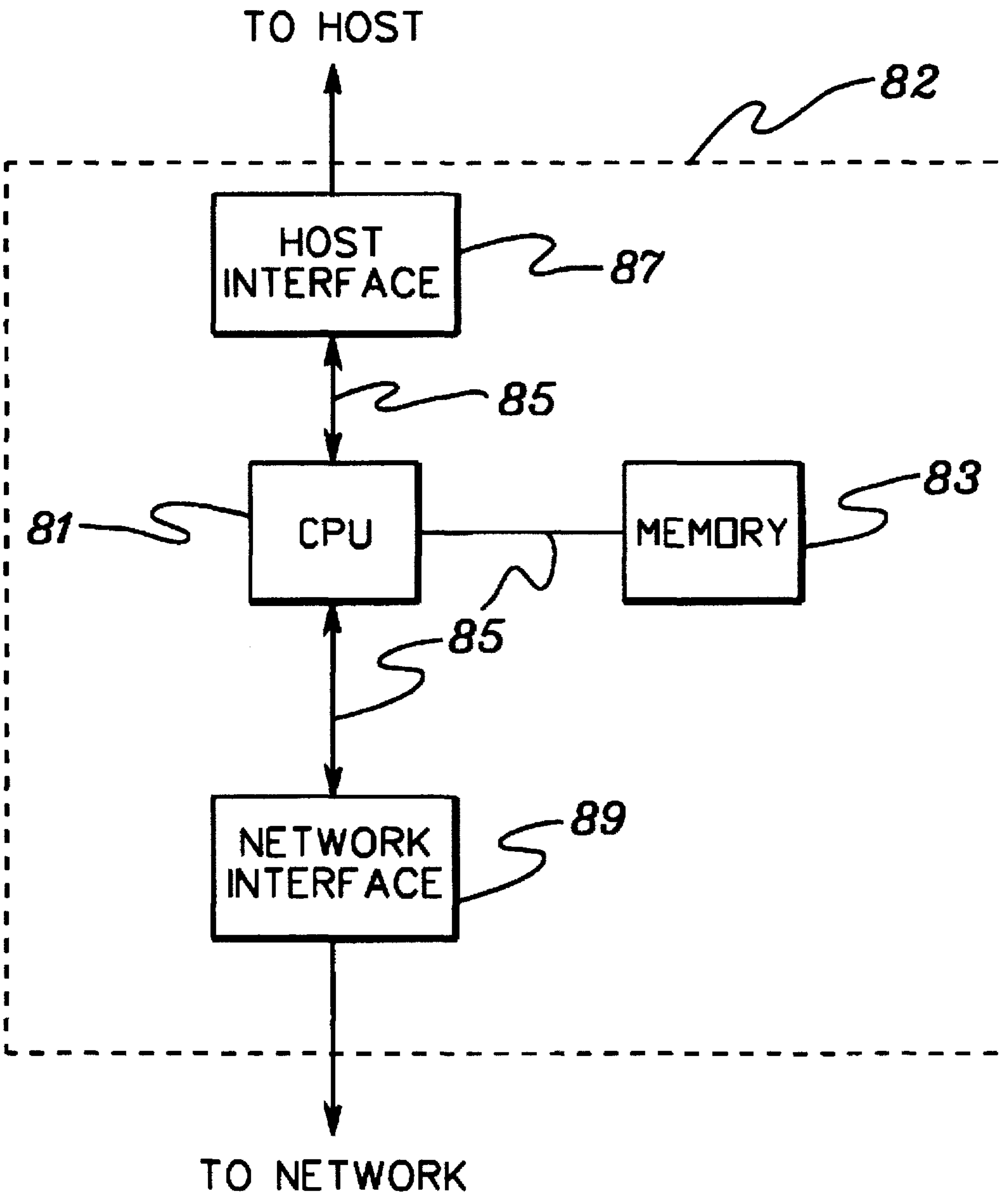

TO HOST
82
HOST INTERFACE
87
85
81
CPU
MEMORY
83
85
NETWORK INTERFACE
89
TO NETWORK

fig. 4

INITIALIZATION SEQUENCE

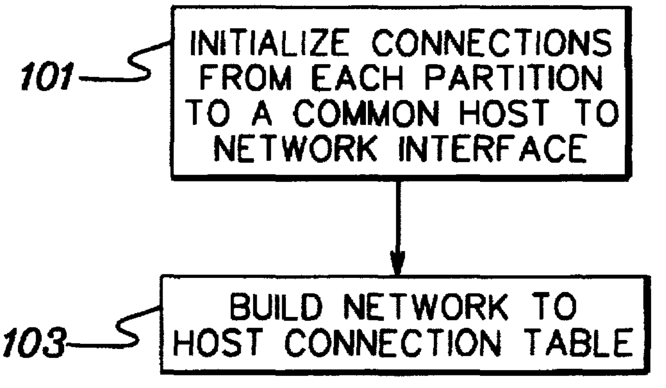


fig. 5

OPERATIONS – OUTBOUND TRAFFIC

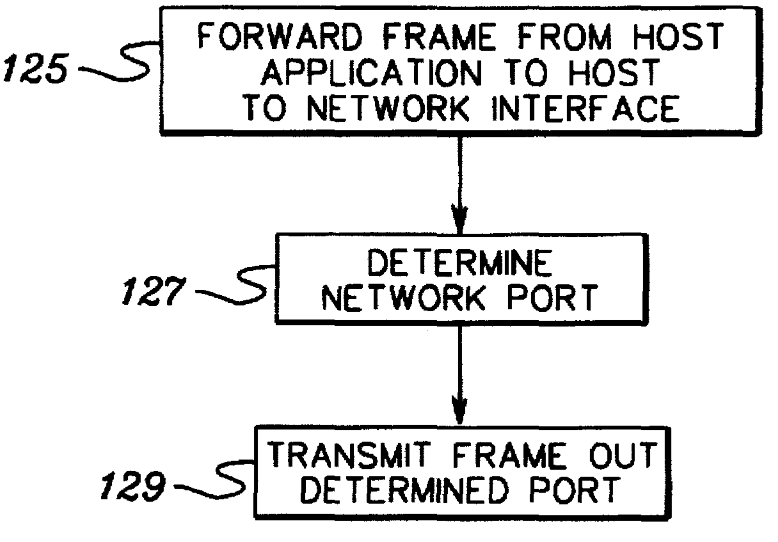
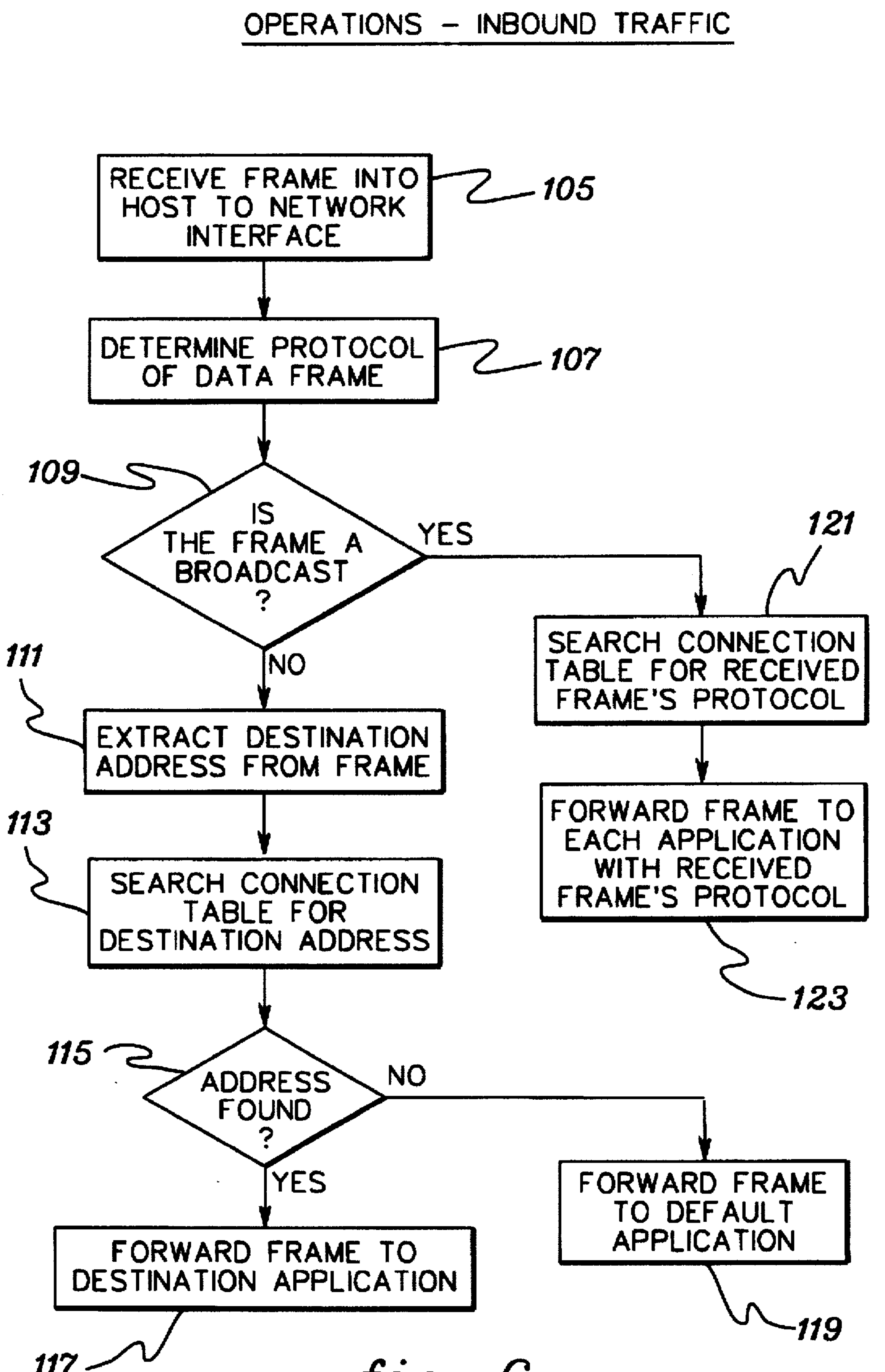


fig. 7



OPERATIONS – INBOUND TRAFFIC
RECEIVE FRAME INTO HOST TO NETWORK INTERFACE
105
DETERMINE PROTOCOL OF DATA FRAME
107
109
IS THE FRAME A BROADCAST ?
YES
NO
121
SEARCH CONNECTION TABLE FOR RECEIVED FRAME'S PROTOCOL
FORWARD FRAME TO EACH APPLICATION WITH RECEIVED FRAME'S PROTOCOL
123
111
EXTRACT DESTINATION ADDRESS FROM FRAME
113
SEARCH CONNECTION TABLE FOR DESTINATION ADDRESS
115
ADDRESS FOUND ?
NO
YES
FORWARD FRAME TO DEFAULT APPLICATION
119
FORWARD FRAME TO DESTINATION APPLICATION
117

fig. 6

METHODS AND SYSTEM FOR NETWORK COMMUNICATIONS OF MULTIPLE PARTITIONS

TECHNICAL FIELD

The present invention relates in general to network communications of processing systems. More particularly, the present invention relates to methods and a system for effecting communications between a network and multiple partitions of a data processing system.

BACKGROUND OF THE INVENTION

Mainframe class data processing systems have hardware and software facilities that enable partitioning thereof. That is, such processing systems may be subdivided into multiple partitions whereby a user of a partition, or software executing in a partition, has the impression that the processing system is exclusively used by them. Each partition has the appearance of being a separate and distinct processing system and may even run its own multitasking and multiuser operating systems independent from each other partition. An IBM Enterprise Systems Architecture ("ESA")/390 mainframe computer is an example of one such partitionable mainframe class data processing system. Partitioning thereof is described in, for example, the publication entitled IBM ESA/390 Principles of Operation, IBM publication number SA22-7201-02, December 1994, and in the IBM Enterprise System/9000 Processor Resource/Systems Manager Planning Guide, IBM publication number GA22-7123-11, April 1994, which are both hereby incorporated herein by reference in their entirety.

Software executing in individual partitions within a mainframe class data processing system may require a network connection such as a Local Area Network ("LAN") connection or a Wide Area Network ("WAN") connection. This may be used to facilitate connectivity to users, or to application programs used in, for example, a client-server processing environment. Shown in FIG. **1** is the conventional configuration used to connect individual partitions, including the software running therein, to a LAN. The configuration includes a processing system **11** that has partitions **13**, **15**, **17**, **19**, **20** and **21**.

Network connectivity for each partition of the system of FIG. **1** is conventionally achieved using separate network interfaces for each partition. For example, partition **13** is connected through channel connection **29** to an IBM 3172 Interconnect Controller 23 (with 8232 Channel Interface Attachment) that has, for example, a token ring LAN port **32** attached to LAN **37** thereby providing LAN connection **31**. Network connectivity is accordingly directly provided between partition **13** and computers **43** and **45** on LAN **37** through the IBM 3172 23. However, according to conventional techniques, this configuration has no other direct logical or physical connections from any of the other partitions to LAN **37**. To further note, each application within partition **23** must communicate with a different network port on IBM 3172 23. The IBM 3172 (having internal 8232 Channel Interface Attachment), is described in a publication entitled 8232 LAN Channel Station, Apr. 15, 1988, IBM publication number ZZ25-8577-0, that is incorporated herein by reference in its entirety.

The conventional software executing on IBM 3172s restricts direct logical connectivity to being between a single partition and its corresponding LAN. Thus, to facilitate direct connectivity from a computer **47** on a LAN **39** to both partitions **17** and **21**, multiple IBM 3172s must be used. Partition **17** is coupled to LAN **39** via channel connection **29'**, IBM 3172 25 and LAN port **34** thereby establishing LAN connection **33**. Similarly, partition **21** is coupled to LAN **39** via channel connection **29"**, IBM 3172 27, and LAN port **35** thereby establishing LAN connection **36**.

The conventional host to network connectivity techniques have several limitations. Connectivity between a single network (i.e. LAN or WAN) and multiple partitions require the use of multiple interfaces therebetween such as, for example, multiple IBM 3172s. Further, each application executing in a single partition must use a different port on the IBM 3172 corresponding to the single partition. Of course, using multiple interfaces and ports introduces further problems including requirements for additional cabling, floor space, power and cooling. Systems management also becomes more complex as each of the multiple interfaces must be managed separately. Even further, costs associated with the required software licenses for each of the multiple interfaces may become excessive. The present invention is directed toward solutions for the above noted problems.

DISCLOSURE OF THE INVENTION

In a first aspect, the present invention includes, a method of network communications for use in a mainframe class data processing system having multiple partitions and a port to a network. The method includes establishing a table defining communications paths between the port to the network and at least two partitions of the multiple partitions. Data frames are passed between the network and the at least two partitions of the multiple partitions. Specifically, the data frames are passed through the port to the network and along the communications paths defined in the table such that network communications is effected.

As an enhancement, each partition of the at least two partitions may have at least one application executing therein. The establishing step may then comprise defining within the table each application of the at least one application executing with each partition of the at least two partitions to have a logical application address associated with it. The passing step then may comprise extracting a logical destination address from a received data frame, and if a logical application address corresponding to the logical destination address is in the table, then passing the received data frame to a destination application defined within the table and having its logical application address corresponding to the logical destination address.

In another aspect, the present invention includes a method of network communications for use in a mainframe class data processing system having multiple partitions, a port to a network, and a table. Each partition of the at least two partitions has at least one application executing therein. Further, the table contains an application definition associated with the port to the network for the at least one application executing within each partition. Each application definition comprises a logical address.

The method includes receiving an inbound data frame having a logical destination address from the network through the port to the network. The logical destination address is then extracted from the inbound data frame and a destination application is looked up in the table, the destination application having a corresponding logical address to the logical destination address. If the destination application is found in the table, then the inbound data frame is passed to the destination application such that network communication is effected between the network and the destination application.

As an enhancement, the mainframe class data processing system may have multiple ports to multiple networks. Each application definition may then include a port parameter such that the receiving step of the method comprises receiving the inbound data frame from a receiving port of the multiple ports. The looking up step may then comprise looking up the destination application in application definitions of the table wherein the port parameter corresponds to the receiving port.

As a further enhancement, the method may include transmitting an outbound data frame from a transmitting application of the at least one application executing within each partition. Transmitting the outbound data frame comprises determining an outbound port of the multiple ports of the system that corresponds to the transmitting application within the table. The outbound data frame is then sent out of the outbound port onto an attached network of the multiple networks.

As yet another enhancement, the mainframe class data processing system may have multiple tables that together comprise the application definitions. A first table of the multiple tables comprises a protocol portion of the application definitions and includes at least one protocol and associated pointer to a secondary table of the multiple tables. The secondary table corresponds to the at least one protocol. The extracting step may then comprise extracting a received protocol from the inbound data frame, looking up the received protocol in the first table to obtain the associated pointer to a second table and looking up the destination application in the secondary table.

In a further embodiment, the mainframe class data processing system may have at least one partition with multiple applications executing therein. The table then defines communications paths between the port to the network and at least two applications of the multiple applications. Data frames may then be passed between the network and the at least two applications along the communications paths defined in the table and through the port to the network.

Further enhancements to the above-described methods and a corresponding system for network communications are recited herein.

To summarize, the techniques of the present invention have many advantages and features associated with them. By facilitating network communications from multiple applications within multiple partitions through a common host to network interface, the resources required to perform such network communications have been reduced. Specifically, the hardware resources required have been reduced along with the associated software and license costs. Reduced hardware resources translate into a reduction in required channels, floor space, cabling, cooling, and power. In terms of software maintenance, a single configuration console corresponding to the single host to network interface may be used to manage all system network connections rather than multiple consoles for each separate, for example, IBM 3172, previously required. Further, migration to a system employing the techniques of the present invention may be performed without modifying current application programs. All these advantages and features translate into increased reliability, availability, and serviceability ("RAS") for the processing system.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter regarded as the present invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of practice, together with further objects and advantages thereof, may best be understood by reference to the following detailed description of a preferred embodiment and the accompanying drawings in which:

FIG. 1 is a system diagram of a conventional network connected partitionable mainframe class data processing system;

FIG. 2 is a system diagram of a network connected partitionable mainframe class data processing system pursuant to an embodiment of the present invention;

FIG. 3 is a system diagram of a network connected partitionable mainframe class data processing system in accordance with an alternate embodiment of the present invention;

FIG. 4 is a block diagram of a host to network interface pursuant to an embodiment of the present invention;

FIG. 5 is a flow diagram of an initialization method in conformance with one embodiment of the present invention;

FIG. 6 is a flow diagram of an operational method for processing inbound network traffic pursuant to an embodiment of the present invention; and

FIG. 7 is a flow diagram of an operational method for processing outbound network traffic according to one embodiment of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

Shown in FIG. 2 is a partitionable mainframe class data processing system 11 (e.g., an IBM Enterprise System/9000) having an integral host to network interface ("HNI") 51 that facilitates a LAN connection 55 from multiple partitions 13, 15, 17, 19, 20 and 21 to LAN 53 through LAN port 54. Each application in each partition may directly communicate with computers 61, 63 and 65 on LAN 53 through the single host to network interface 51 and single LAN port 54. The LAN shown is a Token Ring LAN; however, the present invention is equally applicable to other types of LANs such as, for example, Ethernet and Fiber Distributed Data Interface ("FDDI") as will be apparent to one of ordinary skill in the art. Further, different applications within a single partition may communicate through a common network port. The host to network interface may support multiple network connections by way of multiple network ports. For example, a WAN connection 57 comprising, for example, a Peer to Peer Protocol ("PPP") connection is established to a computer 59 through WAN port 56. Any mix of LAN and WAN connections among multiple ports of host to network interface 51 is possible.

Although the host to network interface 51 of FIG. 2 is part of the processor 11, other embodiments of the present invention are possible. As one example, an IBM 3172 67 (FIG. 3) may be loaded with software to enable its use as the HNI of the present invention; the techniques implemented by the software are discussed further hereinbelow. In this example, single HNI 67 provides direct network connectivity for each of partitions 13, 15, 17, 19 and 20 within mainframe class data processing system 11 to computers 71 and 73 on LAN 75. A channel connection 29 comprising, for example, an Enterprise System Connection ("ESCON") connects processing system 11 to HNI 67. LAN 75 connects to a network port 68 of HNI 67 thereby providing LAN connection 69.

Regardless of whether the host to network interface is internal to the processor (FIG. 2-51) or external to the

processor (FIG. 3-67), the HNI 82 (FIG. 4) includes similar elements. A CPU 81 processes the communications traffic passing through the HNI 82. A memory 83 is connected to CPU 81 and facilitates storage of programs and data used for processing. CPU 81 is further connected to a host interface 87 that is connected to a host processor. Host interface 87 may comprise, for example, an ESCON channel interface if the HNI 82 is an IBM 3172, or it may comprise logic that directly connects to internal processor busses if the HNI 82 is disposed within the mainframe processor itself. Further included in HNI 82 is at least one network interface 89 that connects to one of the types of LANs or WANs as discussed hereinabove, while additional network interfaces may be added to HNI 82 as necessary. Each element shown within HNI 82 connects to CPU 81 via bidirectional buses 85. The design and construction of the HNI 82 hardware along with many possible variations thereof will be apparent to one of ordinary skill in the art.

The network communications techniques of the present invention are described hereinbelow with regard to the flow diagrams of FIGS. 5–7. The implementation of each of the individual steps of these techniques within an HNI will be apparent to one of ordinary skill in the art.

Conventionally, an application program within a partition initiating communications across a network will present various initialization commands to the processing system. These commands are detected by the appropriate IBM 3172 by means of an associated device address included within the commands that the IBM 3172 recognizes. As is well known, the device address identifies both the application sending the commands and the partition that the application is executing in. Specifically, the device address is available on the internal processor busses and on ESCON Multiple Image Facility ("EMIF") channels. The device address accompanies all network communications commands sent by an application including initialization commands. Further information regarding this aspect of IBM 3172 operations will be apparent to one of ordinary skill in the art and is described in the IBM 823 LAN Channel Station document incorporated by reference hereinabove.

According to the techniques of the present invention, the initialization sequence of each application remains the same as if it were communicating with a conventional IBM 3172. This sequence comprises, for example, a STARTUP/STARTLAN sequence for Transmission Control Protocol/Internet Protocol ("TCP/IP") communications and is described in, for example, the IBM 8232 LAN Channel Station document incorporated by reference hereinabove and TCP/IP Tutorial and Technical Overview, IBM document number GG24-3376-01, published Jun. 5, 1990 and hereby incorporated by reference herein in its entirety.

According to the techniques of the present invention, a common HNI is responsive to initialization sequences from multiple applications in multiple partitions (FIG. 5-101). When an initialization sequence is detected by the HNI, an entry is added for the initiating application to a network to host connection table, referred to herein as a "connection table" (103). An example of a connection table and description of the fields therein are set forth below:

Device Address	Port	Protocol	Logical Address
1-2	1	TCP/IP	1.2.3.4
1-3	1	TCP/IP	2.3.4.5

-continued

Device Address	Port	Protocol	Logical Address
1-4	1	TCP/IP	3.4.5.6
3-1	2	TCP/IP	4.5.6.7
4-1	2	TCP/IP	5.6.7.8

Device Address—A combination of the partition number that the application is executing within and the application's hardware address within the partition.

Protocol—The networking protocol that the application is using. Examples of networking protocols include TCP/IP, "NETBIOS," "IPX" (used in "Novell" networks) and Systems Network Architecture ("SNA").

Logical Address—A logical address for the application used by the particular networking protocol. For example, an address for a TCP/IP application may comprise 1.2.3.4. In the particular case of TCP/IP, a network mask field may also be included in the table. TCP/IP addressing and network masks will be apparent to one of ordinary skill in the art.

Port—The network port on the HNI that is to be used for communications with the application.

By way of example, the first table entry corresponds to an application executing in partition 1 and having an application hardware address within the partition of 2. The application will communicate through network port 1 using TCP/IP protocol and the logical address of the application is 1.2.3.4 (the network mask for this TCP/IP example is not shown). The second table entry corresponds to another application that is executing in partition 1 and communicating using TCP/IP through port 1, however having an application hardware address of 3 and a TCP/IP address of 2.3.4.5.

After initialization, operational sequences may begin. As one example of an operational sequence, inbound network traffic is processed (FIG. 6). Such processing begins with the receipt of a data frame from a network, through a port and into the HNI (105). A received data frame may have, for example, the following format and content as will be apparent to one of ordinary skill in the art:

MAC	LLC	DATA

MAC—Media Access Control—This field contains, for example, broadcast indications and other LAN specific information.

LLC—Logical Link Control—This field contains, for example, protocol information.

DATA—This field contains, for example, the transferred data at this protocol level.

Once the data frame has been received, the protocol of the frame is extracted from the LLC field thereof (107). As known in the art, the LLC field has the following format and sub-fields:

DSAP	SSAP	CF

DSAP—Destination Service Access Point—This field indicates the logical protocol of the data frame, for example, TCP/IP.

SSAP—Source Service Access Point—This field indicates the logical address of the source of the data frame.

CF—Control Field—This field includes miscellaneous control parameters.

The protocol of the received data frame is extracted from the DSAP field of the LLC.

The MAC field of the received data frame is then examined to determine if the received data frame comprises a broadcast (**109**). If so, a search of the connection table for each application that has the same protocol as that of the received (broadcast) data frame (**121**) is performed. After that, a copy of the frame is forwarded to each application having the same protocol (**123**). Transfer of the broadcast data frame to the destination applications is performed using the device addresses that can be found in the connection table associated with each application. The individual steps of performing such a transfer will be apparent to one of ordinary skill in the art.

If the received data frame is not a broadcast data frame, then further processing is performed. The logical destination address is extracted from the data field of the received data frame (**111**). The position and format of the logical destination address will be apparent to one of ordinary skill in the art based upon the previously determined protocol. The connection table is then searched for an application having the same logical address as the logical destination address of the received data frame (**113**). If the logical destination address is found as a logical address in the table (**115**) then the received data frame is passed to the application (**117**) corresponding to the found logical address. Otherwise, the data frame may be discarded or passed to a default application for processing (**119**).

In specific regard to the searching of table entries, only table entries corresponding to the port that the data frame was received through are searched (e.g., in step **113**). This not only accelerates the search, but permits applications to have the same logical address on different networks (connected to different network ports). To explain, if multiple applications had the same logical address, a search of the table for a particular (e.g., destination) address might result in an ambiguous result comprising multiple applications. However, if the identically addressed applications are associated with different ports and the search is restricted to applications associated with only one port, such ambiguity is removed.

The techniques of the present invention facilitate the processing of outbound network traffic (FIG. **7**). An outbound data frame flows from an application in a partition to an HNI and out a port to a network. Advantageously, according to the techniques of the present invention, multiple applications in multiple partitions may communicate through a common network port on a common HNI.

Outbound traffic processing begins with the passing of an outbound data frame from an application in a partition to the HNI (**125**). The HNI extracts an outbound port parameter passed to it from the sending application (**127**) and the outbound data frame is passed through the port, transmitting the data frame (**129**).

In alternate embodiments of the present invention, the connection table may be restructured in many ways. As one example, the single connection table may be replaced with multiple connection tables. For example, a first table may comprise various protocol entries, with each protocol entry having a corresponding pointer to another table that contains the remainder of the "connection" table information. An example of the fields of a record within the above described first table is set forth below:

Protocol	Pointer to Second Table

The second table may have, for example the following fields:

Device Address	Port	Protocol	Logical Address

Additional fields in the second table could include, for example, the network mask (for TCP/IP protocol) and the type of network port, for example, Token Ring, Ethernet or FDDI.

Operationally, the two table embodiment functions similarly to the single table embodiment. For example, for inbound traffic processing, once the protocol is determined (e.g., FIG. **6**, **107**), the first table is searched for a protocol entry that corresponds to the received data frame's protocol. Once found, the second table pointed to by the pointer corresponding to the found protocol entry is used as the connection table for the inbound traffic processing techniques. In a broadcast situation, for example, the received data frame is sent to each application listed within the pointed to second table because all applications therein comprise the same protocol as the received data frame. During outbound traffic processing, a protocol for the outbound frame is determined from its LLC field and is looked up in the first table. The corresponding pointer field in the first table is used to identify a second table within which the device address of the application sending the outbound data frame is looked up. A port corresponding to the looked up device address is determined and the outbound data frame is transmitted therethrough.

The techniques of the present invention have many advantages and features associated with them. By facilitating network communications from multiple applications within multiple partitions through a common HNI, the resources required to perform such network communications have been reduced as compared to the conventional requirement of separate host to network interfaces for each communicating partition. Specifically, the hardware resources required have been reduced along with the associated software and license costs. Reduced hardware resources translate into a reduction in required channels, floor space, cabling, cooling, and power. In terms of software maintenance, a single configuration console corresponding to the single host to network interface may be used to manage all system network connections rather than multiple consoles for each separate, for example, IBM 3172, previously required. Further, migration to a system employing the techniques of the present invention may be performed without modifying current application programs. All these advantages and features translate into increased reliability, availability, and serviceability ("RAS") for the processing system.

While the invention has been described in detail herein in accordance with certain preferred embodiments thereof, many modifications and changes therein may be effected by those skilled in the art. Accordingly, it is intended by the following claims to cover all such modifications and changes as fall within the true spirit and scope of the invention.

What is claimed is:

1. A method of network communications implemented within a host network interface for use in a mainframe class

data processing system having multiple partitions and a port to a network, said method comprising the steps of:

(a) establishing a table defining communications paths between the port to the network and at least two partitions of the multiple partitions; and

(b) passing data frames between the network and the at least two partitions of the multiple partitions, said passing data frames being through the port to the network and along the communications paths defined in the table established in said step (a) such that said network communications is effected.

2. The method of claim 1, wherein said port to said network comprises one of a Local Area Network ("LAN") port and a Wide Area Network ("WAN") port such that said passing step (b) comprises passing data frames between said one of said LAN port and said WAN port, and said at least two partitions.

3. The method of claim 2, wherein each partition of the at least two partitions has at least one application executing therein, and wherein said establishing step (a) further comprises defining within said table each application of the at least one application executing within each partition of the at least two partitions to have a logical application address associated with it, and wherein said passing step (b) further comprises extracting a logical destination address from a received data frame of said data frames, and if a logical application address corresponding to said logical destination address is in said table then passing said received data frame to a destination application of the applications defined within said table that has its logical application address corresponding to the logical destination address of the received data frame.

4. The method of claim 3, wherein said passing step (b) further comprises passing the received data frame to a default application executing in a partition of said at least two partitions if a logical application address corresponding to said logical destination address of said received data frame is not in said table.

5. The method of claim 3, wherein said establishing step (a) further comprises associating a device address with each application defined within said table, and wherein said passing step (b) further comprises passing said received data frame to said partition within which said destination application is executing along with a device address associated with the destination application in the table.

6. The method of claim 3, wherein said establishing step (a) further comprises associating a protocol with each application defined within said table, and wherein said passing step (b) further comprises checking said received data frame for a broadcast indication, and if said broadcast indication is found then extracting a broadcast frame protocol from said received data frame and passing said received data frame to each application defined in said table to have said broadcast frame protocol as its protocol.

7. The method of claim 2, wherein said establishing step (a) comprises establishing multiple tables, said multiple tables together defining said communications paths between the port to the network and the at least two partitions of the multiple partitions, said passing step (b) being performed according to said communications paths defined in said multiple tables.

8. A method of network communications implemented within a host network interface for use in a mainframe class data processing system having multiple partitions, a port to a network, and a table, each partition of at least two partitions of the multiple partitions having at least one application executing therein, said table containing an application definition associated with the port to the network for the at least one application executing within each partition of the at least two partitions, each application definition comprising a logical address, said method comprising the steps of:

(a) receiving an inbound data frame from said network through said port to said network, said inbound data frame having a logical destination address;

(b) extracting said logical destination address from said inbound data frame and looking up in said table a destination application of the at least one application executing within each partition of the at least two partitions having a corresponding logical address to said logical destination address; and

(c) if said destination application having said corresponding logical address is found in said table, passing said inbound data frame to said destination application such that said network communications is effected between said network and said destination application.

9. The method of claim 8, wherein said mainframe class data processing system further comprises multiple ports to multiple networks, and wherein each application definition further comprises a port parameter such that said receiving step (a) comprises receiving said inbound data frame from a receiving port of said multiple ports, and said looking up step (b) comprises looking up said destination application in application definitions of said table having port parameters corresponding to said receiving port.

10. The method of claim 9, wherein said method further comprises transmitting an outbound data frame from a transmitting application of said at least one application executing within each partition of said at least two partitions, said transmitting comprising determining an outbound port of said multiple ports corresponding to said transmitting application within said table, and sending said outbound data frame out of said outbound port onto an attached network of said multiple networks.

11. The method of claim 8, wherein said extracting step (b) comprises extracting said logical destination address from a data field of said inbound data frame.

12. The method of claim 8, wherein each application definition further comprises a protocol, and wherein said method further comprises checking said inbound data frame for a broadcast indication and if a broadcast indication is present then said extracting step (b) further comprises extracting a broadcast protocol from a Destination Service Access Point ("DSAP") field of said inbound data frame and looking up all applications having a corresponding protocol in said table, and wherein said passing step (c) comprises passing said inbound data frame thereto.

13. The method of claim 8, wherein said mainframe class data processing system has multiple tables that together comprise said application definitions, a first table of said multiple tables comprising a protocol portion of said application definitions and including at least one protocol and an associated pointer to a secondary table of said multiple tables, said secondary table corresponding to said at least one protocol, said extracting step (b) further comprising extracting a received protocol from said inbound data frame, looking up said received protocol in said at least one protocol of said first table to obtain said associated pointer to a pointed to secondary table, and looking up said destination application in said pointed to secondary table.

14. A method of network communications implemented within a host network interface for use in a mainframe class data processing system having at least one partition having multiple applications executing therein, and a port to a network, said method comprising the steps of:

(a) establishing a table defining communications paths between the port to the network and at least two applications of the multiple applications; and

(b) passing data frames between the network and the at least two applications of the multiple applications, said passing data frames being through the port to the network and along the communications paths defined in the table established in said step (a) such that said network communications is effected.

15. A system for network communications in a mainframe class data processing system having at least one partition having multiple applications executing therein, and a port to a network, said system comprising:

a table defining communications paths between at least two applications of the multiple applications and the port to the network;

means for passing data frames between the network and the at least two applications of the multiple applications, said data frames passing through the port to the network and along the communications paths defined in the table such that said network communications is effected; and

wherein said table and said means for passing data frames are disposed within a host to network interface.

16. The system of claim **15**, wherein said mainframe class processing system comprises at least two partitions, and wherein each application of the at least two applications executes within a different partition of the at least two partitions.

17. The system of claim **15**, wherein said system comprises multiple ports to at least one network and wherein said communications paths defined in said table comprise communications paths between the at least two applications of the multiple applications and a first port of the multiple ports to a first network of the at least one network, and at least one communications path between an application of the multiple applications and a second port of the multiple ports to a second network of the at least one network.

18. The system of claim **15**, wherein said port to the network comprises one of a Local Area Network ("LAN") port and a Wide Area Network ("WAN") port.

19. The system of claim **15** wherein said host to network interface is disposed external to said mainframe class data processing system and is connected thereto by a channel type attachment.

20. A host-to-network interface for a partitioned mainframe class data processing system having multiple partitions, comprising:

a network interface for coupling at least two partitions of the multiple partitions to a network;

a memory having a table established therein for defining communication paths between the network interface and at least two partitions of the multiple partitions.

21. The host-to-network interface of claim **20**, wherein the table contains at least one device address associated with at least one of the at least two partitions for establishing a communication path thereto.

22. The host-to-network interface of claim **21**, wherein the at least one device address is further associated with an application within the at least one partition, the communication path being established to said application within the at least one partition.

* * * * *

US006040861A

United States Patent [19]

[11] Patent Number: 6,040,861
[45] Date of Patent: Mar. 21, 2000

Boroczky et al.

[54] **ADAPTIVE REAL-TIME ENCODING OF VIDEO SEQUENCE EMPLOYING IMAGE STATISTICS**

[75] Inventors: **Lilla Boroczky**, Endicott; **Charlene Ann Gebler**, Vestal; **John M. Kaczmarczyk**; **Edward F. Westermann**, both of Endicott; **Robert L. Woodard**, Newark Valley, all of N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/948,442**

[22] Filed: **Oct. 10, 1997**

[51] **Int. Cl.**[7] **H04N 7/18**

[52] **U.S. Cl.** **348/409**; 370/276; 370/278; 382/236; 382/239; 395/800

[58] **Field of Search** 348/409, 404, 348/405, 700, 452, 412, 420

[56] **References Cited**

U.S. PATENT DOCUMENTS

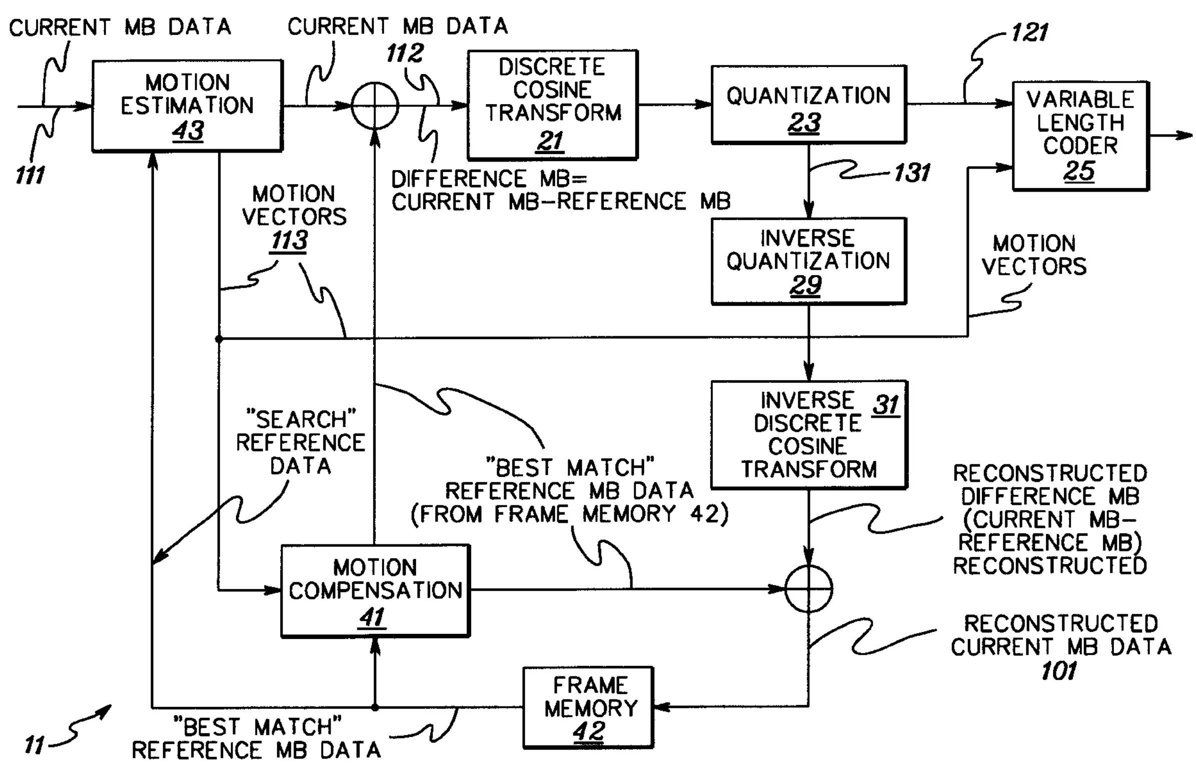
4,395,732	7/1983	Upton	358/169
4,868,653	9/1989	Golin et al.	358/133
5,136,377	8/1992	Johnston et al.	358/136
5,196,930	3/1993	Kadono et al.	358/133
5,321,522	6/1994	Eschbach	358/433
5,400,075	3/1995	Savatier	348/384
5,440,346	8/1995	Alattar et al.	348/420
5,469,212	11/1995	Lee	348/412
5,781,788	7/1998	Woo et al.	370/276
5,793,895	8/1998	Chang et al.	348/402

Primary Examiner—Andy Rao
Assistant Examiner—Shawn An
Attorney, Agent, or Firm—Heslin & Rothenberg, P.C.

[57] **ABSTRACT**

Method, system and computer program product are provided for adaptively encoding in hardware, software or a combination thereof a sequence of video frames in real-time. A first encoding subsystem is employed to analyze the sequence of video frames to derive information on at least one characteristic thereof, such as motion statistics, non-motion statistics, scene change statistics, or scene fade statistics. The gathered information may comprise either an intraframe characteristic or an interframe characteristic. A second encoding subsystem, coupled to the first encoding subsystem, encodes the sequence of video frames employing at least one controllable parameter. The second encoding subsystem dynamically adapts intraframe or interframe encoding of the sequence of video frames by adjusting the at least one controllable parameter used in the encoding process in response to the derived information from the first encoding subsystem.

26 Claims, 6 Drawing Sheets



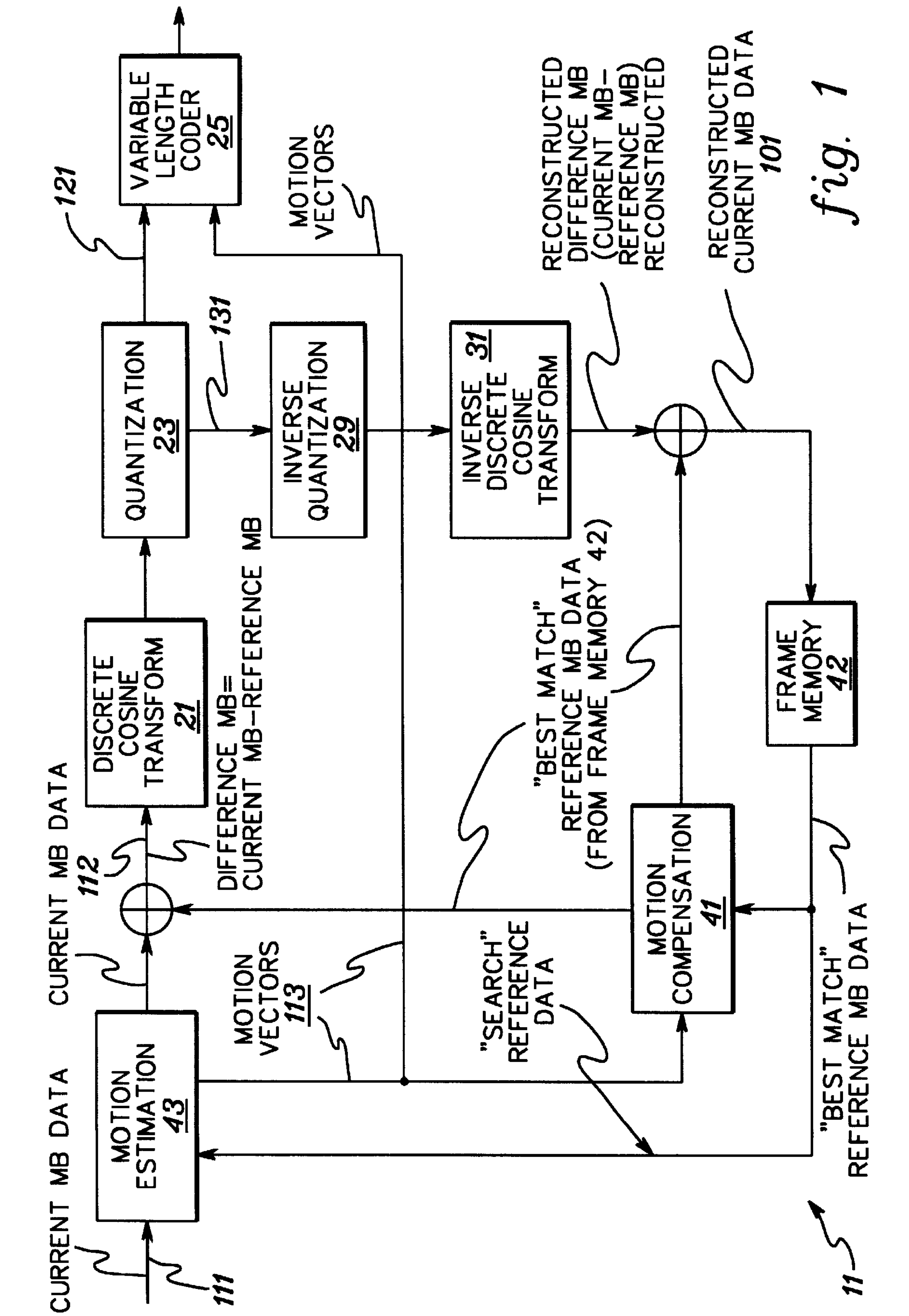

CURRENT MB DATA
111
MOTION ESTIMATION 43
CURRENT MB DATA
112
DISCRETE COSINE TRANSFORM 21
DIFFERENCE MB= CURRENT MB–REFERENCE MB
QUANTIZATION 23
121
VARIABLE LENGTH CODER 25
131
INVERSE QUANTIZATION 29
MOTION VECTORS
MOTION VECTORS 113
"SEARCH" REFERENCE DATA
"BEST MATCH" REFERENCE MB DATA (FROM FRAME MEMORY 42)
INVERSE DISCRETE COSINE TRANSFORM 31
RECONSTRUCTED DIFFERENCE MB (CURRENT MB–REFERENCE MB) RECONSTRUCTED
RECONSTRUCTED CURRENT MB DATA 101
MOTION COMPENSATION 41
FRAME MEMORY 42
"BEST MATCH" REFERENCE MB DATA
11

fig. 1

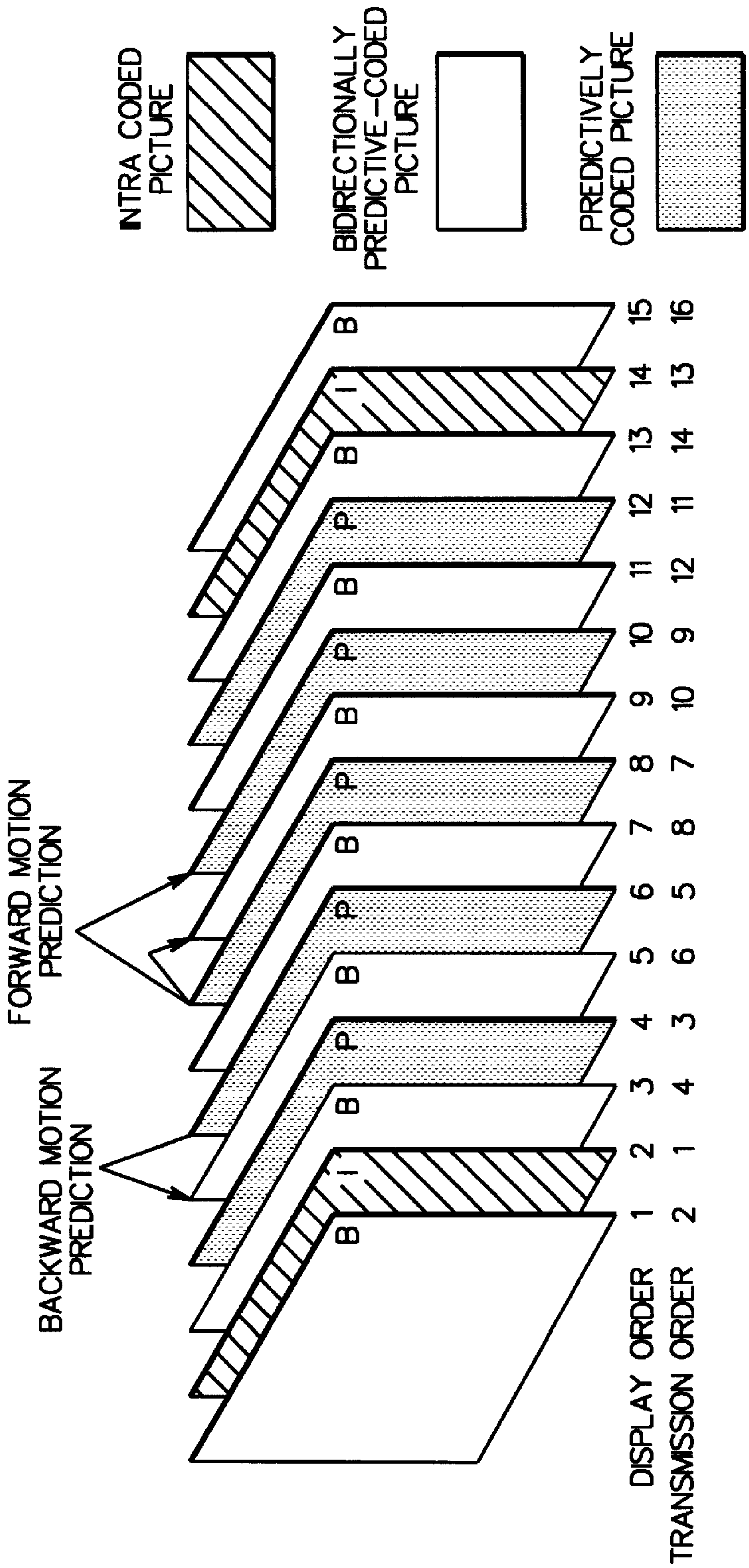
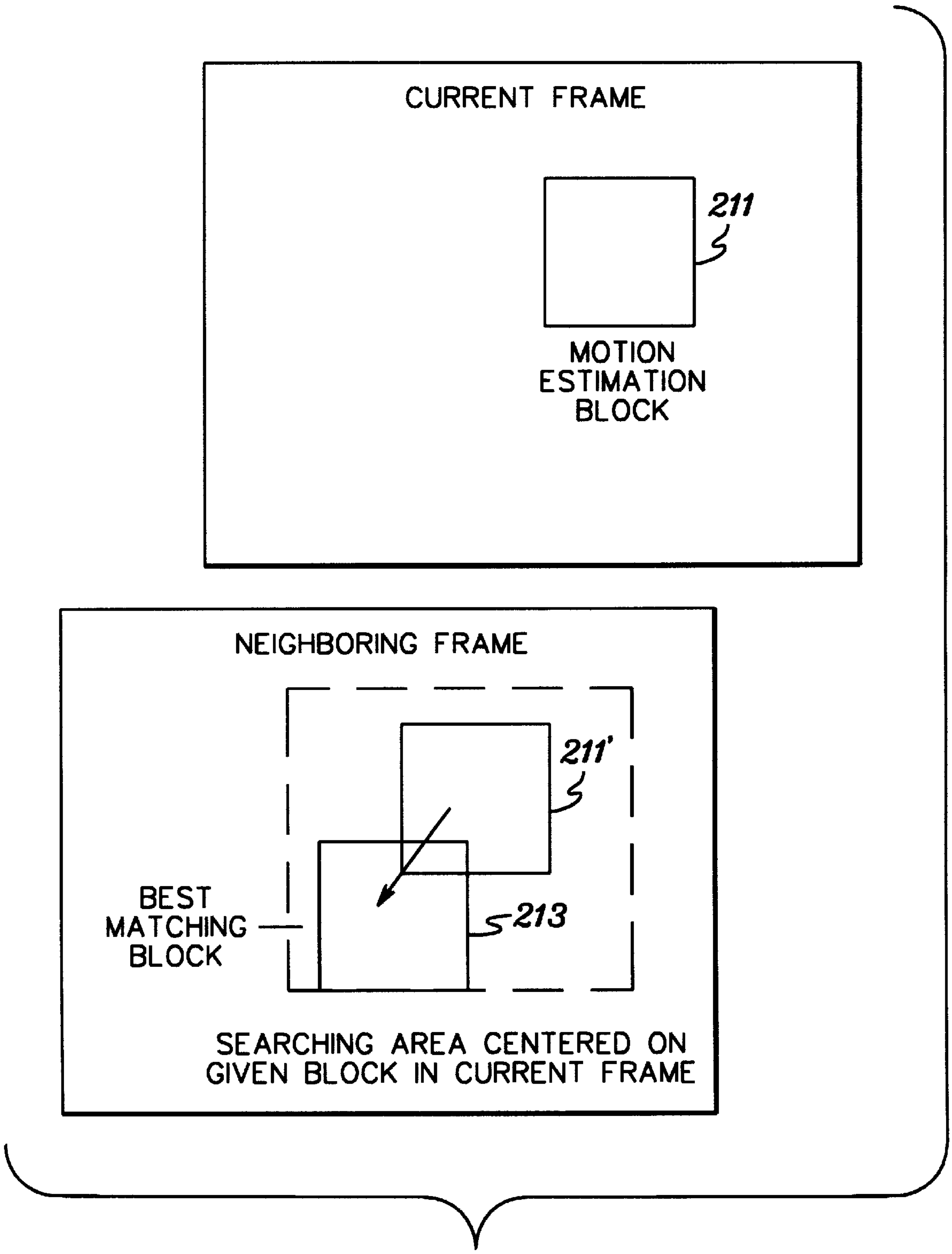
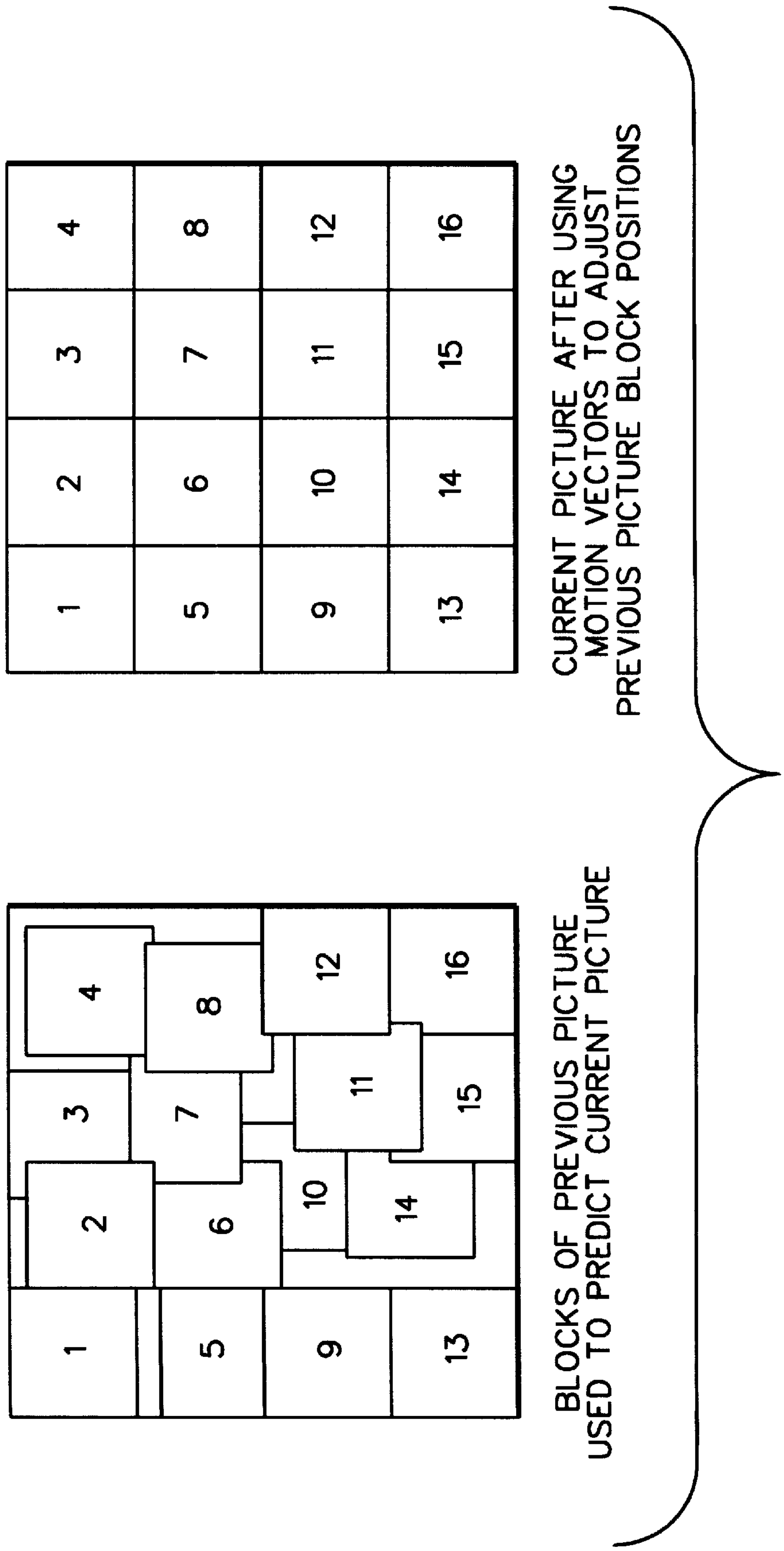

INTRA CODED PICTURE
BIDIRECTIONALLY PREDICTIVE-CODED PICTURE
PREDICTIVELY CODED PICTURE
FORWARD MOTION PREDICTION
BACKWARD MOTION PREDICTION
B I B P B P B P B P B P B I B
DISPLAY ORDER 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
TRANSMISSION ORDER 2 1 4 3 6 5 8 7 10 9 12 11 14 13 16

fig. 2



CURRENT FRAME
211
MOTION ESTIMATION BLOCK
NEIGHBORING FRAME
211'
BEST MATCHING BLOCK
213
SEARCHING AREA CENTERED ON GIVEN BLOCK IN CURRENT FRAME

fig. 3



1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
CURRENT PICTURE AFTER USING MOTION VECTORS TO ADJUST PREVIOUS PICTURE BLOCK POSITIONS
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
BLOCKS OF PREVIOUS PICTURE USED TO PREDICT CURRENT PICTURE

fig. 4

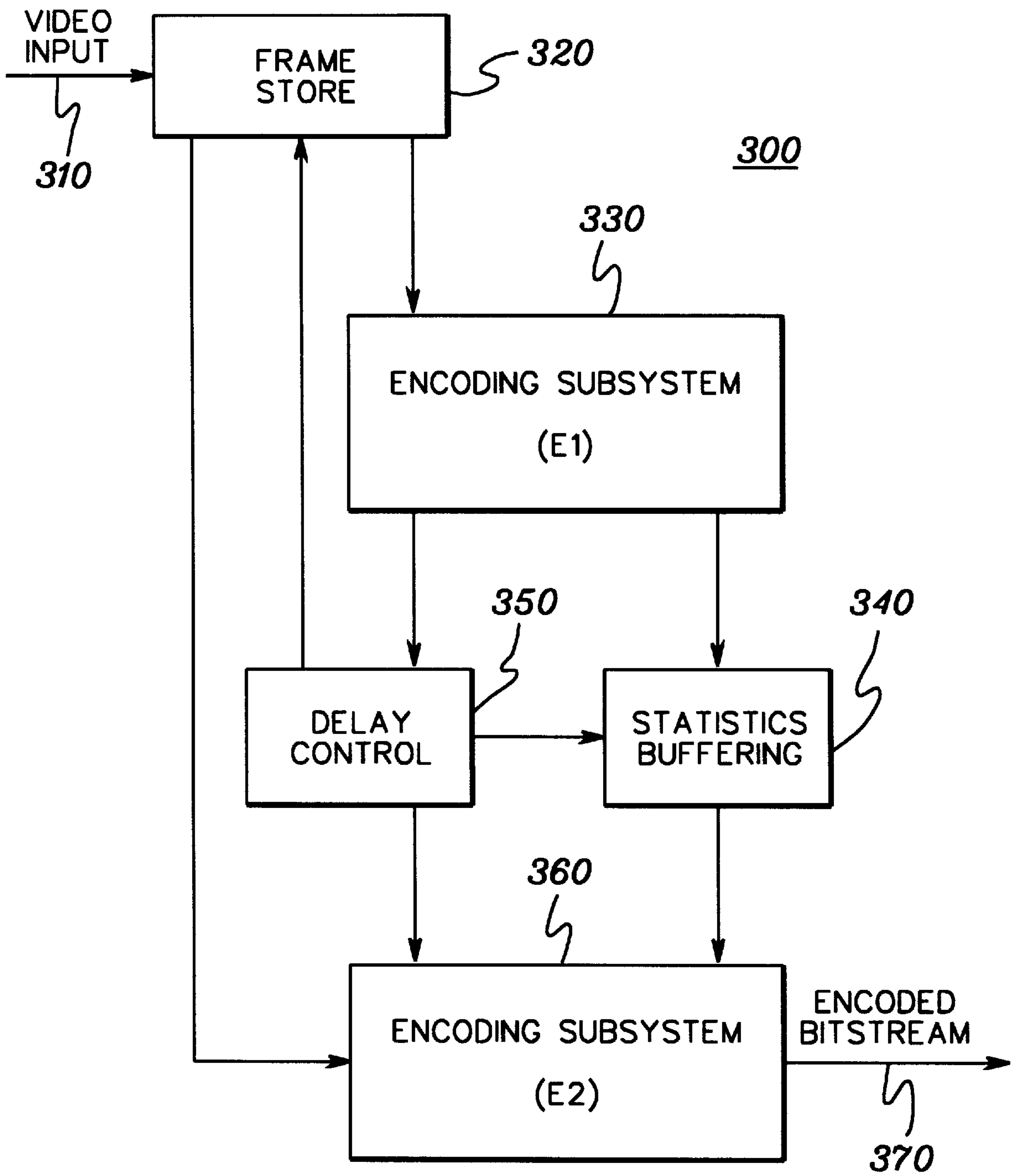

VIDEO INPUT
310
FRAME STORE
320
300
330
ENCODING SUBSYSTEM
(E1)
350
340
DELAY CONTROL
STATISTICS BUFFERING
360
ENCODING SUBSYSTEM
(E2)
ENCODED BITSTREAM
370

fig. 5

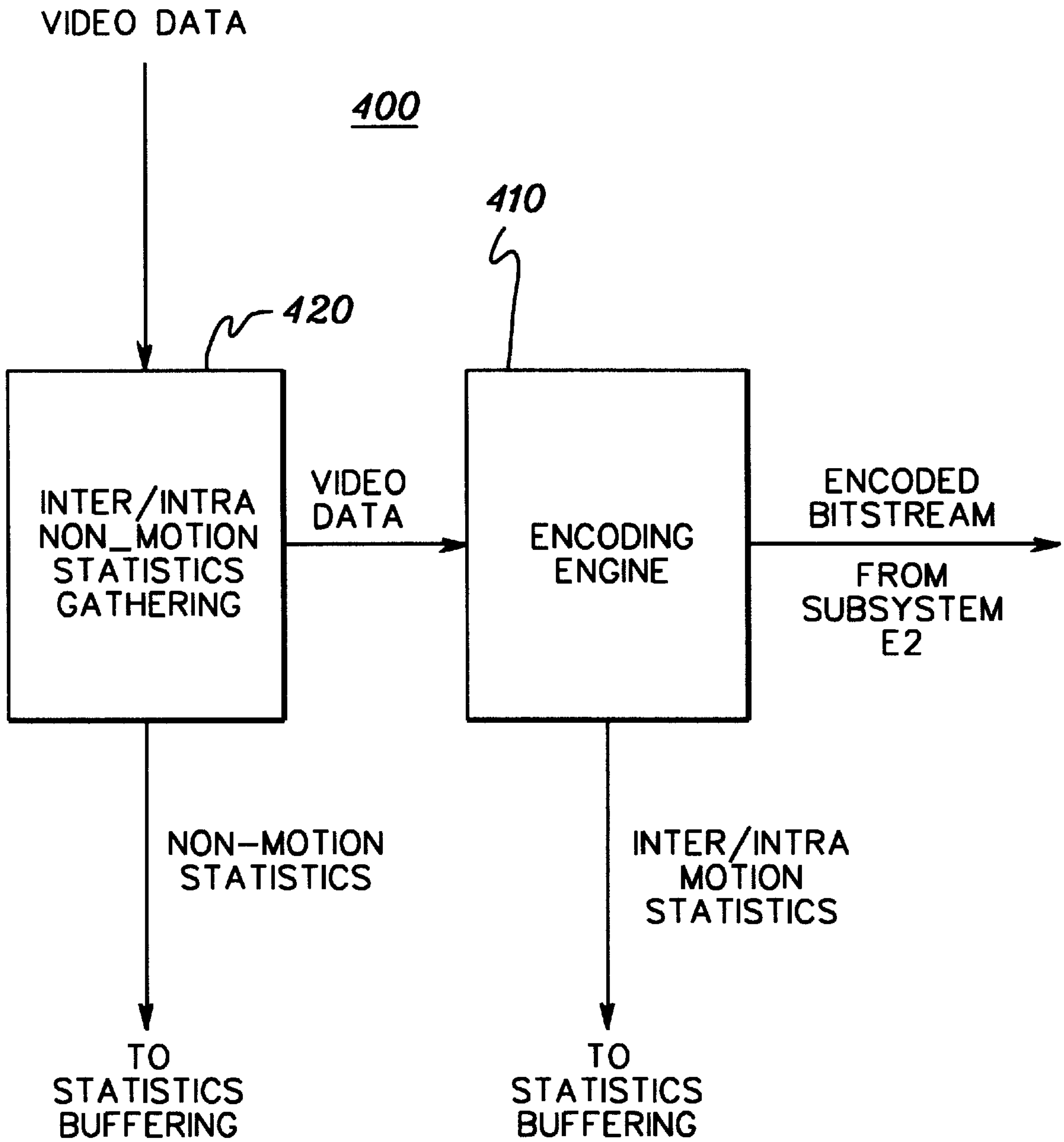

VIDEO DATA
400
410
420
INTER/INTRA NON_MOTION STATISTICS GATHERING
VIDEO DATA
ENCODING ENGINE
ENCODED BITSTREAM
FROM SUBSYSTEM E2
NON-MOTION STATISTICS
INTER/INTRA MOTION STATISTICS
TO STATISTICS BUFFERING
TO STATISTICS BUFFERING

fig. 6

ADAPTIVE REAL-TIME ENCODING OF VIDEO SEQUENCE EMPLOYING IMAGE STATISTICS

TECHNICAL FIELD

This invention relates in general to compression of digital visual images, and more particularly, to a technique for real-time encoding of a video sequence using image statistics derived from the video sequence to dynamically change one or more controllable encoding parameter(s) from frame to frame or within a frame.

BACKGROUND OF THE INVENTION

Within the past decade, the advent of world-wide electronic communications systems has enhanced the way in which people can send and receive information. In particular, the capabilities of real-time video and audio systems have greatly improved in recent years. In order to provide services such as video-on-demand and video conferencing to subscribers, an enormous amount of network bandwidth is required. In fact, network bandwidth is often the main inhibitor in the effectiveness of such systems.

In order to overcome the constraints imposed by networks, compression systems have emerged. These systems reduce the amount of video and audio data which must be transmitted by removing redundancy in the picture sequence. At the receiving end, the picture sequence is uncompressed and may be displayed in real-time.

One example of an emerging video compression standard is the Moving Picture Experts Group ("MPEG") standard. Within the MPEG standard, video compression is defined both within a given picture and between pictures. Video compression within a picture is accomplished by conversion of the digital image from the time domain to the frequency domain by a discrete cosine transform, quantization, and variable length coding. Video compression between pictures is accomplished via a process referred to as motion estimation and compensation, in which a motion vector plus difference data is used to describe the translation of a set of picture elements (pels) from one picture to another.

The ISO MPEG-2 standard specifies only the syntax of bitstream and semantics of the decoding process. The choice of coding parameters and tradeoffs in performance versus complexity are left to the encoder developers.

One aspect of the encoding process is compressing a digital video image into as small a bitstream as possible while still maintaining video detail and quality. The MPEG standard places limitations on the size of the bitstream, and requires that the encoder be able to perform the encoding process. Thus, simply optimizing the bit rate to maintain desired picture quality and detail can be difficult.

For example, a bit rate is defined in bits per second. Based on the frame rate and type of picture being encoded, a number of bits per picture is assigned. At 6,000,000 bits per second (6 Mbps), and pictures at 30 picture frames per second, each picture would be allocated 200,000 bits assuming that the bits are allocated uniformly. With a 720×480 picture having 1350 macroblocks, this translates into 148 bits allocated per macroblock. Thus, in the case of scene changes and action videos, the bit rate can be quickly consumed with drastic changes between macroblocks and/or between frames. Picture quality and detail can suffer as a result.

In view of the above-noted constraints, therefore, this invention seeks to enhance picture quality of an encoded video sequence while still obtaining a high compression rate by providing an adaptive real-time encoding scheme.

DISCLOSURE OF THE INVENTION

Briefly summarized, this invention comprises in one aspect a method for encoding a sequence of video frames. The method includes: analyzing the sequence of video frames to derive information on at least one characteristic thereof, the at least one characteristic comprising at least one of an intraframe characteristic or an interframe characteristic; encoding the sequence of video frames employing at least one controllable parameter; and dynamically adapting the encoding of the sequence of video frames using the information on the at least one characteristic thereof to adjust the at least one controllable parameter employed in encoding the sequence of video frames. Advantageously, the encoding is dynamically adaptive to change in the at least one characteristic of the sequence of video frames.

In another aspect, the invention comprises a system for encoding a sequence of video frames. The system includes means for analyzing the sequence of video frames to derive information on at least one characteristic thereof. The at least one characteristic comprises at least one of an intraframe characteristic or an interframe characteristic. The system further includes means for encoding the sequence of video frames employing at least one controllable parameter, and means for dynamically adapting the encoding of the sequence of frames using the derived information on the at least one characteristic thereof. The information is used to adjust the at least one controllable parameter employed during encoding the sequence of video frames such that the encoding process is dynamically adaptive to change in the at least one characteristic of the sequence of video frames being encoded.

In a further aspect, the invention comprises a computer program product including a computer usable medium having computer readable program code means therein for use in encoding a sequence of video frames. The computer readable program product means in the computer program product includes computer readable program code means for causing a computer to affect: analyzing of the sequence of video frames to derive information on at least one characteristic thereof, the at least one characteristic comprising at least one of an intraframe characteristic or an interframe characteristic; encoding of the sequence of video frames employing at least one controllable parameter; and dynamically adapting the encoding of the sequence of video frames using the information on the at least one characteristic thereof to adjust the at least one controllable parameter employed in encoding the sequence of video frames.

In general, encoding in accordance with the principles of the present invention results in improved picture quality compared with non-adaptive encoder systems, especially at low bit rates. This is because, for example, employing adaptive bit allocation among frames (as well as within frames) is more critical in low bit rate encoding compared with higher bit rate encoding. Further, the encoding technique of this invention can insure a semi-constant picture quality of a decoded video sequence in constant bit rate (CBR) mode or a constant picture quality in variable bit rate (VBR) encoding mode.

BRIEF DESCRIPTION OF THE DRAWINGS

The above-described objects, advantages and features of the present invention, as well as others, will be more readily understood from the following detailed description of cer-

tain preferred embodiments of the invention, when considered in conjunction with the accompanying drawings in which:

FIG. **1** shows a flow diagram of a generalized MPEG-2 compliant encoder **11**, including a discrete cosine transformer **21**, a quantizer **23**, a variable length coder **25**, an inverse quantizer **29**, an inverse discrete cosine transformer **31**, motion compensation **41**, frame memory **42**, and motion estimation **43**. The data paths include the i^{th} picture input **111**, difference data **112**, motion vectors **113** (to motion compensation **41** and to variable length coder **25**), the picture output **121**, the feedback picture for motion estimation and compensation **131**, and the motion compensated picture **101**. This figure has the assumptions that the i^{th} picture exists in frame memory or frame store **42** and that the $i+1^{th}$ is being encoded with motion estimation.

FIG. **2** illustrates the I, P, and B pictures, examples of their display and transmission orders, and forward, and backward motion prediction.

FIG. **3** illustrates the search from the motion estimation block in the current frame or picture to the best matching block in a subsequent or previous frame or picture. Elements **211** and **211**' represent the same location in both pictures.

FIG. **4** illustrates the movement of blocks in accordance with the motion vectors from their position in a previous picture to a new picture, and the previous picture's blocks adjusted after using motion vectors.

FIG. **5** shows a flow diagram of an encoding system **300** employing a first encoding subsystem **E1** and a second encoding subsystem **E2** in accordance with the principles of the present invention. Subsystem **E1** is configured to derive statistics on one or more characteristics of a sequence of frames to be encoded. These characteristics are employed by subsystem **E2** to adaptively encode the sequence of frames to optimize picture quality and/or encoding performance.

FIG. **6** is a generalized diagram of an encoding subsystem in accordance with the present invention. Subsystem **E1** is employed to generate, for example, non-motion statistics and inter/intra motion statistics using non-motion statistics gathering **420** and encoding engine **410**, respectively, while subsystem **E2** generates the encoded bitstream using encoding engine **410**.

BEST MODE FOR CARRYING OUT THE INVENTION

The invention relates, for example, to MPEG compliant encoders and encoding processes such as described in "Information Technology-Generic coding of moving pictures and associated audio information: Video," Recommendation ITU-T H.262, ISO/IEC 13818-2, Draft International Standard, 1994. The encoding functions performed by the encoder include data input, spatial compression, motion estimation, macroblock type generation, data reconstruction, entropy coding, and data output. Spatial compression includes discrete cosine transformation (DCT), quantization, and entropy encoding. Temporal compression includes intensive reconstructive processing, such as inverse discrete cosine transformation, inverse quantization, and motion compensation. Motion estimation and compensation are used for temporal compression functions. Spatial and temporal compression are repetitive functions with high computational requirements.

More particularly the invention relates, for example, to a process for performing spatial and temporal compression including discrete cosine transformation, quantization, entropy encoding, motion estimation, motion compensation, and prediction, and even more particularly to a system for accomplishing spatial and temporal compression.

The first compression step is the elimination of spatial redundancy, for example, the elimination of spatial redundancy in a still picture of an "I" frame picture. Spatial redundancy is the redundancy within a picture. The MPEG-2 Draft Standard is using a block based method of reducing spatial redundancy. The method of choice is the discrete cosine transformation, and discrete cosine transform coding of the picture. Discrete cosine transform coding is combined with weighted scalar quantization and run length coding to achieve desirable compression.

The discrete cosine transformation is an orthogonal transformation. Orthogonal transformations, because they have a frequency domain interpretation, are filter bank oriented. The discrete cosine transformation is also localized. That is, the encoding process samples on an 8×8 spatial window which is sufficient to compute 64 transform coefficients or sub-bands.

Another advantage of the discrete cosine transformation is that fast encoding and decoding algorithms are available. Additionally, the sub-band decomposition of the discrete cosine transformation is sufficiently well behaved to allow effective use of psychovisual criteria.

After transformation, many of the frequency coefficients are zero, especially the coefficients for high spatial frequencies. These coefficients are organized into a zig-zag or alternate-scanned pattern, and converted into run-amplitude (run-level) pairs. Each pair indicates the number of zero coefficients and the amplitude of the non-zero coefficient. This is coded in a variable length code.

Motion compensation is used to reduce or even eliminate redundancy between pictures. Motion compensation exploits temporal redundancy by dividing the current picture into blocks, for example, macroblocks, and then searching in previously transmitted pictures for a nearby block with similar content. Only the difference between the current block pels and the predicted block pels extracted from the reference picture is actually compressed for transmission and thereafter transmitted.

The simplest method of motion compensation and prediction is to record the luminance and chrominance, i.e., intensity and color, of every pixel in an "I" picture, then record changes of luminance and chrominance, i.e., intensity and color for every specific pixel in the subsequent picture. However, this is uneconomical in transmission medium bandwidth, memory, processor capacity, and processing time because objects move between pictures, that is, pixel contents move from one location in one picture to a different location in a subsequent picture. A more advanced idea is to use a previous or subsequent picture to predict where a block of pixels will be in a subsequent or previous picture or pictures, for example, with motion vectors, and to write the result as "predicted pictures" or "P" pictures. More particularly, this involves making a best estimate or prediction of where the pixels or macroblocks of pixels of the i^{th} picture will be in the $i-1^{th}$ or $i+1^{th}$ picture. It is one step further to use both subsequent and previous pictures to predict where a block of pixels will be in an intermediate or "B" picture.

To be noted is that the picture encoding order and the picture transmission order do not necessarily match the picture display order. See FIG. **2**. For I-P-B systems the input picture transmission order is different from the encoding order, and the input pictures must be temporarily stored until used for encoding. A buffer stores this input until it is used.

For purposes of illustration, a generalized flowchart of MPEG compliant encoding is shown in FIG. **1**. In the flowchart the images of the i^{th} picture and the $i+1^{th}$ picture are processed to generate motion vectors. The motion vectors predict where a macroblock of pixels will be in a prior and/or subsequent picture. The use of the motion vectors is a key aspect of temporal compression in the MPEG standard. As shown in FIG. **1** the motion vectors, once generated, are used for the translation of the macroblocks of pixels, from the i^{th} picture to the $i+1^{th}$ picture.

As shown in FIG. **1**, in the encoding process, the images of the i^{th} picture and the $i+1^{th}$ picture are processed in the encoder **11** to generate motion vectors which are the form in which, for example, the $i+1^{th}$ and subsequent pictures are encoded and transmitted. An input image **111** of a subsequent picture goes to the motion estimation unit **43** of the encoder. Motion vectors **113** are formed as the output of the motion estimation unit **43**. These vectors are used by the motion compensation Unit **41** to retrieve macroblock data from previous and/or future pictures, referred to as "reference" data, for output by this unit. One output of the motion compensation Unit **41** is negatively summed with the output from the motion estimation unit **43** and goes to the input of the Discrete Cosine Transformer **21**. The output of the discrete cosine transformer **21** is quantized in a quantizer **23**. The output of the quantizer **23** is split into two outputs, **121** and **131**; one output **121** goes to a downstream element **25** for further compression and processing before transmission, such as to a run length encoder; the other output **131** goes through reconstruction of the encoded macroblock of pixels for storage in frame memory **42**. In the encoder shown for purposes of illustration, this second output **131** goes through an inverse quantization **29** and an inverse discrete cosine transform **31** to return a lossy version of the difference macroblock. This data is summed with the output of the motion compensation unit **41** and returns a lossy version of the original picture to the frame memory **42**.

As shown in FIG. **2**, there are three types of pictures. There are "Intra pictures" or "I" pictures which are encoded and transmitted whole, and do not require motion vectors to be defined. These "I" pictures serve as a reference image for motion estimation. There are "Predicted pictures" or "P" pictures which are formed by motion vectors from a previous picture and can serve as a reference image for motion estimation for further pictures. Finally, there are "Bidirectional pictures" or "B" pictures which are formed using motion vectors from two other pictures, one past and one future, and can not serve as a reference image for motion estimation. Motion vectors are generated from "I" and "P" pictures, and are used to form "P" and "B" pictures.

One method by which motion estimation is carried out, shown in FIG. **3**, is by a search from a macroblock **211** of an i^{th} picture throughout a region of the next picture to find the best match macroblock **213**. Translating the macroblocks in this way yields a pattern of macroblocks for the $i+1^{th}$ picture, as shown in FIG. **4**. In this way the i^{th} picture is changed a small amount, e.g., by motion vectors and difference data, to generate the $i+1^{th}$ picture. What is encoded are the motion vectors and difference data, and not the $i+1^{th}$ picture itself. Motion vectors translate position of an image from picture to picture, while difference data carries changes in chrominance, luminance, and saturation, that is, changes in shading and illumination.

Returning to FIG. **3**, we look for a good match by starting from the same location in the i^{th} picture as in the $i+1^{th}$ picture. A search window is created in the i^{th} picture. We search for a best match within this search window. Once found, the best match motion vectors for the macroblock are coded. The coding of the best match macroblock includes a motion vector, that is, how many pixels in the y direction and how many pixels in the x direction is the best match displaced in the next picture. Also encoded is difference data, also referred to as the "prediction error", which is the difference in chrominance and luminance between the current macroblock and the best match reference macroblock.

The operational functions of an MPEG-2 encoder are discussed in detail in commonly assigned, co-pending U.S. patent application Ser. No. 08/831,157, by Carr et al., filed Apr. 1, 1997, entitled "Control Scheme For Shared-Use Dual-Port Predicted Error Array," which is hereby incorporated herein by reference in its entirety.

As noted initially, encoder performance and/or picture quality may be enhanced in accordance with the principles of this invention through real-time adaptive video encoding. The video encoder is constructed to be adaptive to the video data received as a sequence of frames. In accordance with one embodiment of this invention, two encoding subsystems are employed. A significant advantage of using two encoding subsystems is the ability to analyze the video sequence prior to its real-time encoding. Analysis of the video sequence comprises calculating one or more statistics which can be derived from the video data.

The statistical measures can describe different characteristics of an image frame, for example, busyness of a frame, motion between image frames, scene change or fading, etc. Using the calculated statistics, adaptive encoding of the video sequence is then carried out by controlling one or more encoding parameters of the real-time encoding process. For example, bit allocation, quantization parameter(s), encoding mode, etc., can be changed from frame to frame or macroblock to macroblock within a given frame according to derived statistics of a characteristic (e.g., scene content) of the particular frame(s).

One embodiment of an encoding system, generally denoted **300**, in accordance with the principles of this invention is depicted in FIG. **5**. The MPEG Standard is again assumed herein for purposes of explanation; however, those skilled in the art will understand that other implementations and standards can employ the adaptive encoding concepts of this invention. System **300** includes two encoder subsystems, designated E**1** **330** and E**2** **360**. In one implementation, encoder subsystems E**1** and E**2** are assumed to have identical hardware, but different software as described hereinbelow. E**1** is programmed to generate the desired statistics, such as interframe/intraframe non-motion, motion, etc. statistics, which are important to the encoding subsystem's (E**2**) specific bit rate control algorithm. E**2** generates encoded frames based on the statistics generated by encoding subsystem E**1**.

Operationally, a sequence of video frames **310** is initially received into a frame store **320**, where one or more frames are buffered depending upon the encoding specification (e.g., I, IP, IBP, IBBP encoding). This is accomplished by partitioning frame store **320** into an appropriate number of picture buffers (determined by group of picture (GOP) structure). These partitions are managed by a delay control logic **350**. After sufficient delay, again determined by implementation, the video frame information is passed to encoder subsystem E**1** **330**, which derives the information on image statistics and stores this information in a statistics buffer **340** on a frame-by-frame basis. The delay control hardware **350** manages buffering of incoming video data and of image statistics, and feeds the video frames from frame

store **320**, as well as the derived statistics from statistics buffering **340**, to encoding subsystem E**2** **360** in encode order. Using these statistics, subsystem E**2** adaptively encodes the frames as described further below and outputs the encoded bitstream **370** in real time, delayed only by sufficient frame time to allow encoding subsystem E**1** to generate the statistics on one or more characteristics of the received video input **310**.

A block diagram of a generalized encoding subsystem, denoted **400**, is depicted in FIG. **6**. System **400** comprises hardware/software for calculating non-motion statistics **420** on the video data as it is received in the encoding subsystem, as well as an encoding engine **410** which consists of hardware/software to perform the actual video compression, i.e., motion estimation, motion compensation, quantization, variable length coding, etc. Encoding subsystem E**1** **330** (FIG. **5**) will employ both statistics gathering logic **420** and encoding engine **410**, while encoding subsystem E**2** **360** (FIG. **5**) will only employ encoding engine **410**. Thus, during a first pass of encoding, i.e., via subsystem E**1**, motion statistics based on motion vectors are calculated by encoding engine **410**. Encoding subsystem E**2** then outputs an encoded bitstream using a second pass through encoding engine **410**.

Real time operation, and associated frame delays of a system in accordance with this invention, is demonstrated in the MPEG-2 example of Table 1. In this example, one B picture is assumed between two anchor pictures (IBPBPBP . . .) and non-motion statistics are being collected. Only one frame of video data is buffered before the statistical calculation, and the delay between input and output of the frames is a maximum of four frame times in the example.

Frame statistics can generally be divided into two groups, i.e., intraframe and interframe statistics. Intraframe statistics are calculated using only pixel data within a frame, while interframe statistics are obtained using several consecutive images from an image sequence (generally two consecutive images). Intraframe and interframe statistics can be further divided into global and local quantities. Global quantities describe characteristics of entire image frames, and local statistical values are calculated for each subdivision of a frame, e.g., for each macroblock of a frame. The statistics presented herein can be calculated from luminance data, however, additional statistics can be derived from chrominance data as well.

Intraframe Statistics

Global Quantities

Average Interpixel Difference (AID) of a frame—AID is the average of absolute differences between two consecutive pixels in the image lines of a frame. The higher the AID, the higher the details in a frame. This quantity is calculated by hardware in the pixel interface as pixels for the frame pass through.

Average Activity (AVACT) of a frame—AVACT is estimated as the average of the macroblock variances in a frame. A macroblock variance can be estimated as statistical averages from the pixel values of a macroblock. The average activity gives information about the busyness of a picture.

Local Quantities

During the calculation of the above global quantities, local measures can also be obtained. Thus, for each subdivision, e.g. macroblock (MB), the MB-AID and MB-AVACT statistics can be stored and then used by subsystem E**2** for local adaptive encoding of a frame.

TABLE 1

Input	Buffer1	Buffer2	Buffer3	Buffer4	Buffer5	Encoder #1	Encoder #2	
n	n	—	—	—	—	—	—	
n + 1	n	n + 1	—	—	—	n	—	
n + 2	n	n + 1	n + 2	—	—	n + 1	—	
n + 3	n	n + 1	n + 2	n + 3	—	n + 2	n	(I)
n + 4	n	n + 1	n + 2	n + 3	n + 4	n + 3	n + 2	(P)
n + 5	n + 5	n + 1	n + 2	n + 3	n + 4	n + 4	n + 1	(B)
n + 6	n + 5	n + 6	n + 2	n + 3	n + 4	n + 5	n + 4	(P)
n + 7	n + 5	n + 6	n + 7	n + 3	n + 4	n + 6	n + 3	(B)
n + 8	n + 5	n + 6	n + 7	n + 8	n + 4	n + 7	n + 6	(P)
n + 9	n + 5	n + 6	n + 7	n + 8	n + 9	n + 8	n + 5	(B)

Other implementations will be apparent to those skilled in the art employing the principles of the present invention. For example, video data could be input into a frame store in parallel with inputting it into the first encoding subsystem E**1** if the video data has no B pictures and the statistics are intraframe.

Examples of statistics calculated in encoding subsystem E**1** are next described.

As noted, encoding subsystem E**1** calculates statistics from the image data. Based upon these statistics, the subsystem can also carry out preprocessing steps, such as identifying scene change or fade detection. The particular statistics calculated by subsystem E**1** depend on the given implementation of a rate control algorithm within subsystem E**2**. In MPEG-2 encoding, there is a wide range of picture statistics that can be used to determine the quantization for a frame or within a frame. The statistics discussed hereinbelow are provided by way of example only and other E**2** video compression algorithms may employ different picture quantities.

Interframe Statistics

These statistics describe the relationship between consecutive frames of an image sequence, e.g. occurrence of motion, scene change, fading, or identifying noise in a macroblock.

Global Quantities

Average Frame Difference (AFD) of a frame—AFD is the average of absolute differences between the luminance pixel value of the current picture and the pixel at the same location in the previous frame.

Variance of the DFD (Displaced Frame Difference)—To obtain DFD at a pixel, motion vectors have to be calculated for each macroblock using two consecutive frames. DFD is the difference between a pixel value at the current frame and the corresponding (displaced by an estimated motion vector) pixel value in the previous or future frame. Variance can be estimated as statistical average of DFDs, which are calculated for each pixel of the entire frame.

Local Quantities

Variance of DFD for macroblock. Same as the global quantity, but the variance is estimated for a macroblock.

Variance of motion vectors (difference between neighboring motion vectors)

Scene Change Detection

Consider two consecutive frames. The above statistics (global and local) are available for each frame. There are different possibilities to detect scene changes; for example:

a) If ((AVACT(i)−AVACT(i−1))>threshold**1**), then frame i belongs to the new scene. Threshold**1** is determined experimentally.

b) If ((AID(i)−AID(i−1))>threshold**2**) then frame i belongs to the new scene. Threshold**2** is determined experimentally.

c) Combination of conditions a) and b).

d) If ((AFD)>threshold**3**), then frame i belongs to the new scene. Threshold**3** is determined experimentally.

e) Scene change detection—interframe statistics. If (DFD variance>threshold**4**) and if ((AID(i)−AID(i−1)) >threshold**2**), then frame i belongs to the new scene. Thresholds are determined experimentally. Threshold**4** has to be chosen carefully, because only one motion vector is available for a macroblock (MB) and it is not necessarily the true motion vector for each pixel in the MB.

If a scene change is detected, this will be communicated to system E**2**. Subsystem E**2** may react by disregarding any information from the previous picture which belongs to the previous scenes. E**2** can also rearrange the encoding modes of the pictures: e.g., the first picture that belongs to the new scene may be coded as an I picture.

Fade Detection

A fade is basically a slow scene change, whereby the frames change gradually, in contrast to a true scene cut where the change is abrupt. A fade has two directions, 1) the first scene dissolving into the fade, and 2) the fade crystallizing into the second scene.

The presence of a fade and its direction may be determined by a percentage of interpixel sums delta with respect to a previous frame, i.e., if the value of the sum of the pixels of frame N+1 differs by an experimentally determined percentage from frame N, then a fade is occurring. Its direction is determined by the sign of the magnitude, i.e., being greater or less than zero. If a fade is detected, this is communicated to E**2**.

As noted above, encoder subsystem E**2** can have the same encoding engine architecture as encoder subsystem E**1**, however, the statistics gathering hardware/software would not be employed by subsystem E**2**. Adaptive encoding of a frame sequence is carried out by the rate control algorithm of the encoding engine, i.e., using the above-generated statistics. This is a 2-step process.

At first, bit allocation is defined for each picture depending on the bit rate, encoding mode and the relative characteristics of the frames to each other. Then, a corresponding quantization parameter (QUANT) is defined. In an MPEG-2 compatible bitstream, the QUANT value can change from macroblock to macroblock allowing locally adaptive quantization inside a frame. In accordance with this invention, a first global QUANT value is defined for each picture using the global quantities described above. The QUANT value for a particular macroblock is then obtained by modulating the global QUANT value based on the local statistics of the macroblock.

By detecting scene change within encoding subsystem E**1**, and with a prior knowledge of picture statistics, after detection of a scene change information from previous pictures which belongs to the previous scene can be discarded. For example, a new group of pictures (GOP) can be started with the new scene. The global QUANT values can be calculated for the frames of the new GOP using predefined initial rate control parameters instead of using parameters from the old scene.

If a fade is detected by subsystem E**1**, then subsystem E**2** may react by using the proper reference frames for motion estimation/compensation, and/or change the encoding mode of frames. An example of this may be forcing an I picture, or coding all macroblocks in a P or B picture as intra-macroblocks.

Locally adaptive quantization is also possible. The aim in this approach would be to distribute allocated bits among the macroblocks based on scene content. One possible approach would be to employ AVACT of the current picture to be encoded. The MB-AVACT for each macroblock are also stored in the statistics buffering. A QUANT value of an MB is obtained by modulating the global QUANT value by the ratio of the AVACT and the MB-AVACT. For example, if the macroblock activity is high with respect to the picture AVACT, then the QUANT of this particular macroblock will be increased with respect to the global QUANT, and vice versa.

Pursuant to the adaptive encoding system of this invention, the local statistics can also be employed to identify noisy macroblocks within a picture. If a macroblock has a high DFD value and its activity is also high, the macroblock can be declared noisy. A noisy macroblock can then be encoded by a high QUANT value, which results in bit savings. These extra bits can be used for encoding noiseless macroblocks within the frame. Consequently, the overall quality of the picture will be improved.

One example of a manner in which the encoding system of this invention performs the function of locally adaptive quantization may be the following. To determine the quantization value for each macroblock, the encoding subsystem would receive the MB-AVACT and DFD from the statistics store **340** (FIG. **5**) for that macroblock. Subsystem E**2** would then use these values in determining the most efficient QUANT value for the macroblock.

Those skilled in the art will note from the above discussion that encoding in accordance with the principles of the present invention results in improved picture quality compared with non-adaptive encoder systems, especially at low bit rates. This is because employing adaptive bit allocation among frames, as well as within frames, is more critical in low bit rate encoding compared with higher bit rate encoding. Further, the encoding technique of this invention can insure a semi-constant picture quality of a decoded video sequence in constant bit rate (CBR) mode or a constant picture quality in variable bit rate (VBR) encoding mode.

The present invention can be included, for example, in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The articles manufactured can be included as part of the computer system or sold separately.

The flow diagrams depicted herein are provided by way of example. There may be variations to these diagrams or the steps or operations described herein without departing from the spirit of the invention. For instance, in certain cases the

steps may be performed in differing order, or steps may be added, deleted or modified. All these variations are considered to comprise part of the present invention as recited in the appended claims.

While the invention has been described in detail herein in accordance with certain preferred embodiments thereof, many modifications and changes therein may be affected by those skilled in the art. Accordingly, it is intended by the appended claims to cover all such modifications and changes as fall within the true spirit and scope of the invention.

We claim:

1. A method for encoding a sequence of video frames comprising:

analyzing the sequence of video frames to derive information on at least one characteristic thereof, said at least one characteristic comprising at least one of an intraframe characteristic or an interframe characteristic;

encoding the sequence of video frames employing at least one controllable parameter;

buffering the sequence of video frames and controlling timing of said encoding of the sequence of video frames so that for each frame of at least some frames of said sequence of video frames said analyzing precedes said encoding to derive said information on the at least one characteristic relative each frame of said at least some frames of the sequence of video frames prior to said encoding of the frame; and

dynamically adapting said encoding of the sequence of video frames using said information on the at least one characteristic thereof to adjust said at least one controllable parameter employed in encoding the sequence of video frames, wherein said encoding is dynamically adaptive to change in the at least one characteristic of the sequence of video frames.

2. The method of claim **1**, wherein said analyzing comprises dynamically analyzing the sequence of video frames to derive statistics on said at least one characteristic thereof, said statistics comprising at least one of motion statistics, non-motion statistics, scene change statistics, or scene fade statistics.

3. The method of claim **2**, wherein said analyzing comprises deriving said statistics from at least one of luminance data and chrominance data of the sequence of video frames.

4. The method of claim **2**, wherein said statistics comprise either global statistics or local statistics, said global statistics comprising statistics on an entire video frame, and said local statistics comprising statistics on a subdivision, picture region, slice, macroblock or block within a video frame of the sequence of video frames.

5. The method of claim **4**, wherein said analyzing comprises determining intraframe statistics for a video frame, said intraframe statistics comprising at least one of an average interpixel difference (AID) of the video frame or an average activity (AVACT) of the video frame.

6. The method of claim **4**, wherein said analyzing comprises determining interframe statistics, said interframe statistics comprising at least one of an average frame difference (AFD), or a variance of a displaced frame difference (DFD).

7. The method of claim **2**, wherein upon said analyzing deriving scene change statistics, said encoding further comprises either disregarding information from a previous frame belonging to a previous scene or rearranging encoding modes of the sequence of video frames.

8. The method of claim **1**, wherein said buffering comprises buffering the sequence of video frames within a frame store and said controlling comprises controlling timing of said encoding of the sequence of video frames so that for each frame of said sequence of video frames said analyzing precedes said encoding to derive said information on the at least one characteristic relative to each frame of the sequence of video frames prior to said encoding of the frame.

9. The method of claim **1**, wherein said analyzing comprises employing a first encoding subsystem to derive said information on said at least one characteristic of the sequence of video frames, and said encoding comprises employing a second encoding subsystem to encode the sequence of video frames using said at least one controllable parameter, said second encoding subsystem being coupled to receive said information from said first encoding subsystem, said dynamically adapting comprising employing said information on the at least one characteristic of the sequence of video frames to dynamically control the at least one controllable parameter employed by said second encoding subsystem in encoding the sequence of video frames.

10. The method of claim **1**, wherein said at least one controllable parameter employed by said encoding comprises at least one of bit allocation, a quantization parameter, or an encoding mode.

11. A system for encoding a sequence of video frames comprising:

means for analyzing the sequence of video frames to derive information on at least one characteristic thereof, said at least one characteristic comprising at least one of an intraframe characteristic or an interframe characteristic;

means for encoding the sequence of video frames employing at least one controllable parameter;

means for buffering the sequence of video frames and means for controlling timing of said encoding of said sequence of video frames so that for each frame of at least some frames of said sequence of video frames said analyzing precedes said encoding to derive said information on the at least one characteristic relative to each frame of said at least some frames of the sequence of video frames prior to said encoding of said frame; and

means for dynamically adapting said encoding of the sequence of video frames using said derived information on the at least one characteristic thereof to adjust said at least one controllable parameter employed in encoding the sequence of video frames, wherein said encoding is dynamically adaptive to change in the at least one characteristic of the sequence of video frames.

12. The system of claim **11**, wherein said means for analyzing comprises a first encoding subsystem, and wherein said means for encoding and said means for dynamically adapting comprise a second encoding subsystem, said second encoding subsystem comprising means for encoding the sequence of video frames in real-time.

13. The system of claim **12**, wherein said first encoding subsystem comprises interframe and intraframe non-motion statistics gathering logic and an encoding engine, said statistics gathering logic being for gathering non-motion statistics on the sequence of video frames and said encoding engine being for deriving interframe and intraframe motion statistics on the sequence of video frames.

14. The system of claim **12**, wherein said means for buffering comprises a frame store for receiving the sequence of video frames, and said means for controlling comprises delay control logic for controlling output of the sequence of video frames from the frame store to the first encoding subsystem and to the second encoding subsystem so that for

each frame said first encoding subsystem derives said information on said at least one characteristic prior to said second encoding subsystem beginning encoding of said frame.

15. The system of claim **14**, further comprising a statistics buffer coupled between said first encoding subsystem and said second encoding subsystem, said first encoding subsystem comprising means for outputting said information on said at least one characteristic of said sequence of video frames to said statistics buffer, and said second encoding subsystem comprising means for receiving said information on said at least one characteristic of said sequence of video frames from said statistics buffer.

16. The system of claim **12**, wherein said first encoding subsystem comprises means for dynamically analyzing the sequence of video frames to derive statistics on said at least one characteristic thereof, said statistics comprising at least one of motion statistics, non-motion statistics, scene change statistics, or scene fade statistics.

17. The system of claim **16**, wherein said first encoding subsystem comprises means for deriving said statistics from at least one of luminance data and chrominance data of the sequence of video frames.

18. The system of claim **16**, wherein said statistics comprise intraframe statistics for a video frame, said intraframe statistics comprising at least one of an average interpixel difference (AID) of the video frame or an average activity (AVACT) of the video frame.

19. The system of claim **16**, wherein said statistics comprise interframe statistics, said interframe statistics comprising at least one of an average frame difference (AFD), or a variance of a displaced frame difference (DFD).

20. The system of claim **11**, wherein said at least one controllable parameter employed by said means for encoding comprises at least one of bit allocation, a quantization parameter, or an encoding mode.

21. The system of claim **11**, wherein said means for encoding comprises an MPEG compliant encoding engine.

22. A computer program product comprising a computer usable medium having computer readable program code means therein for use in encoding a sequence of video frames, said computer readable program code means in said computer program product comprising:

computer readable program code means for causing a computer to affect analyzing of the sequence of video frames to derive information on at least one characteristic thereof, said at least one characteristic comprising at least one of an intraframe characteristic or an interframe characteristic;

computer readable program code means for causing a computer to affect encoding the sequence of video frames employing at least one controllable parameter;

computer readable program code means for causing a computer to affect timing control over said analyzing and said encoding such that encoding of each frame of at least some frames of said sequence of video frames follows analyzing said frame of said at least some frames of said sequence of video frames to derive said information on the at least one characteristic relative to said frame before encoding of said frame; and

computer readable program code means for causing a computer to affect dynamically adapting said encoding of the sequence of video frames using said information on the at least one characteristic thereof to adjust said at least one controllable parameter employed in encoding the sequence of video frames, wherein said encoding is dynamically adaptive to change in the at least one characteristic of the sequence of video frames.

23. The computer readable program code means of claim **22**, wherein said computer readable program code means for causing a computer to affect analyzing of the sequence of frames comprises computer readable program code means for causing a computer to affect dynamically analyzing the sequence of video frames to derive statistics on the at least one characteristic thereof, said statistics comprising at least one of motion statistics, non-motion statistics, scene change statistics, or scene fade statistics.

24. The computer readable program code means of claim **22**, wherein said computer readable program code means for causing a computer to affect analyzing of the sequence of video frames comprises computer readable program code means for causing a computer to affect deriving statistics on said at least one characteristic of said sequence of video frames from at least one of luminance data and chrominance data of said sequence of video frames.

25. The computer readable program code means of claim **22**, wherein said computer readable program code means for causing a computer to affect timing control comprises computer readable program code means for causing a computer to affect timing control over said analyzing and said encoding such that encoding of each frame of said sequence of video frames follows analyzing said frame of said sequence of video frames to derive said information on the at least one characteristic relative to said frame before encoding of said frame.

26. The computer readable program code means of claim **22**, wherein said at least one controllable parameter comprises at least one of bit allocation, a quantization parameter, or an encoding mode.

* * * * *

US005978916A

United States Patent [19]

Randall

[11] **Patent Number: 5,978,916**

[45] **Date of Patent: Nov. 2, 1999**

[54] **METHOD, SYSTEM AND COMPUTER PROGRAM PRODUCT FOR UPDATING REGION-DEPENDENT SOFTWARE USING A COMMON UPDATE MODULE FOR MULTIPLE REGIONS**

[75] Inventor: **Grayson Warren Randall**, Vestal, N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/978,304**

[22] Filed: **Nov. 25, 1997**

[51] **Int. Cl.**[6] **G06F 13/00**

[52] **U.S. Cl.** **713/200**; 395/712; 709/221

[58] **Field of Search** 713/1, 200; 709/221; 395/712; 380/3–5

[56] **References Cited**

U.S. PATENT DOCUMENTS

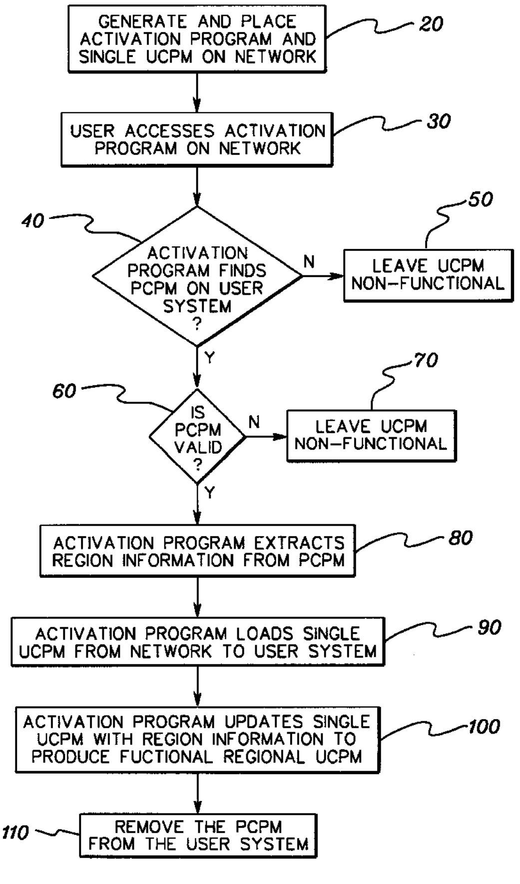
4,634,807	1/1987	Chorley et al.	178/22.08
4,644,493	2/1987	Chandra et al.	364/900
4,866,769	9/1989	Karp	380/4
4,903,296	2/1990	Chandra et al.	380/4
5,008,814	4/1991	Mathur	709/221
5,109,413	4/1992	Comerford et al.	380/4
5,155,847	10/1992	Kirouac et al.	709/221
5,247,683	9/1993	Holmes et al.	709/221
5,359,730	10/1994	Marron	395/712
5,390,297	2/1995	Barber et al.	395/200
5,418,853	5/1995	Kanota et al.	380/5
5,432,647	7/1995	Tateishi	360/60
5,467,396	11/1995	Schossow et al.	380/4
5,513,260	4/1996	Ryan	380/3
5,577,232	11/1996	Priem et al.	709/302
5,732,275	3/1998	Kullick et al.	395/712

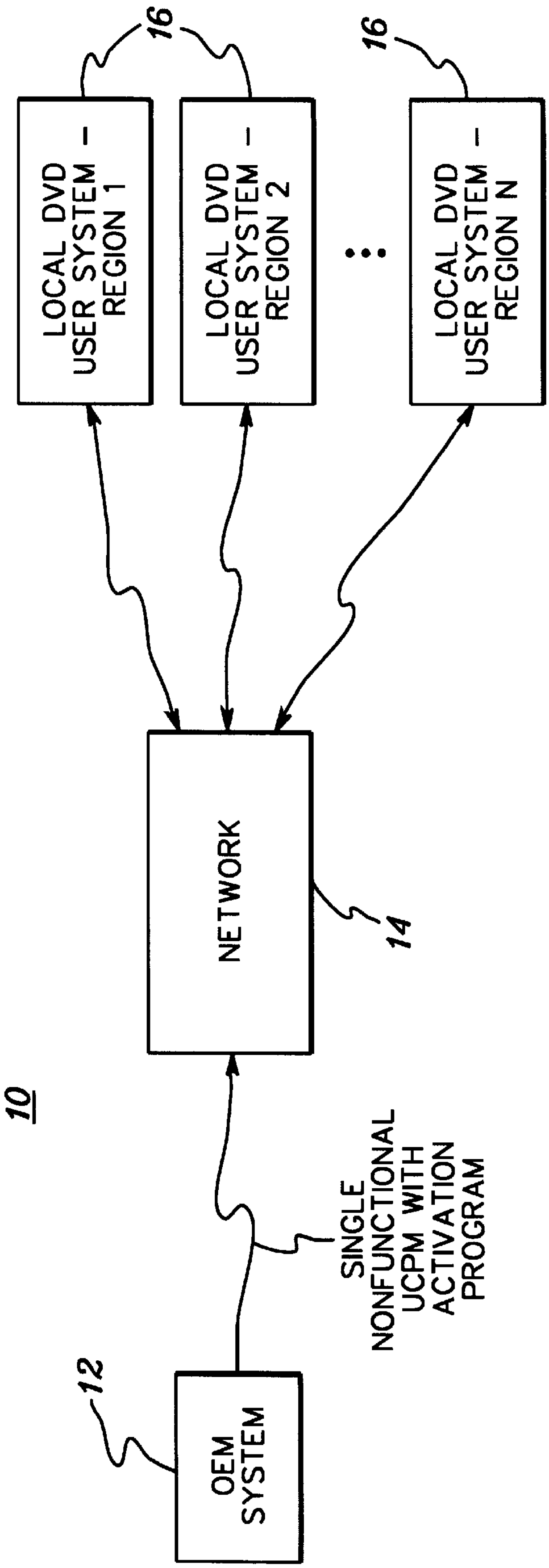
Primary Examiner—Thomas M. Heckler
Attorney, Agent, or Firm—Heslin & Rothenberg, P.C.

[57] **ABSTRACT**

Method, system and computer program product for updating pre-existing region-dependent software within multiple regions via a common software update without affecting the region-dependent nature of the software. The technique includes extracting region information from pre-existing region-dependent software on a user system; loading the non-functional software update onto the user system, the non-functional update being a common software update for the multiple regions; and merging the region information extracted from the pre-existing region-dependent software with the non-functional software update loaded onto the user system to produce a functional, updated region-dependent software on the user system. In one example, the user system comprises a digital video disc (DVD) PC system, and the distribution network is the internet. An activation module accompanies the common software update and accomplishes the extracting, loading and merging steps.

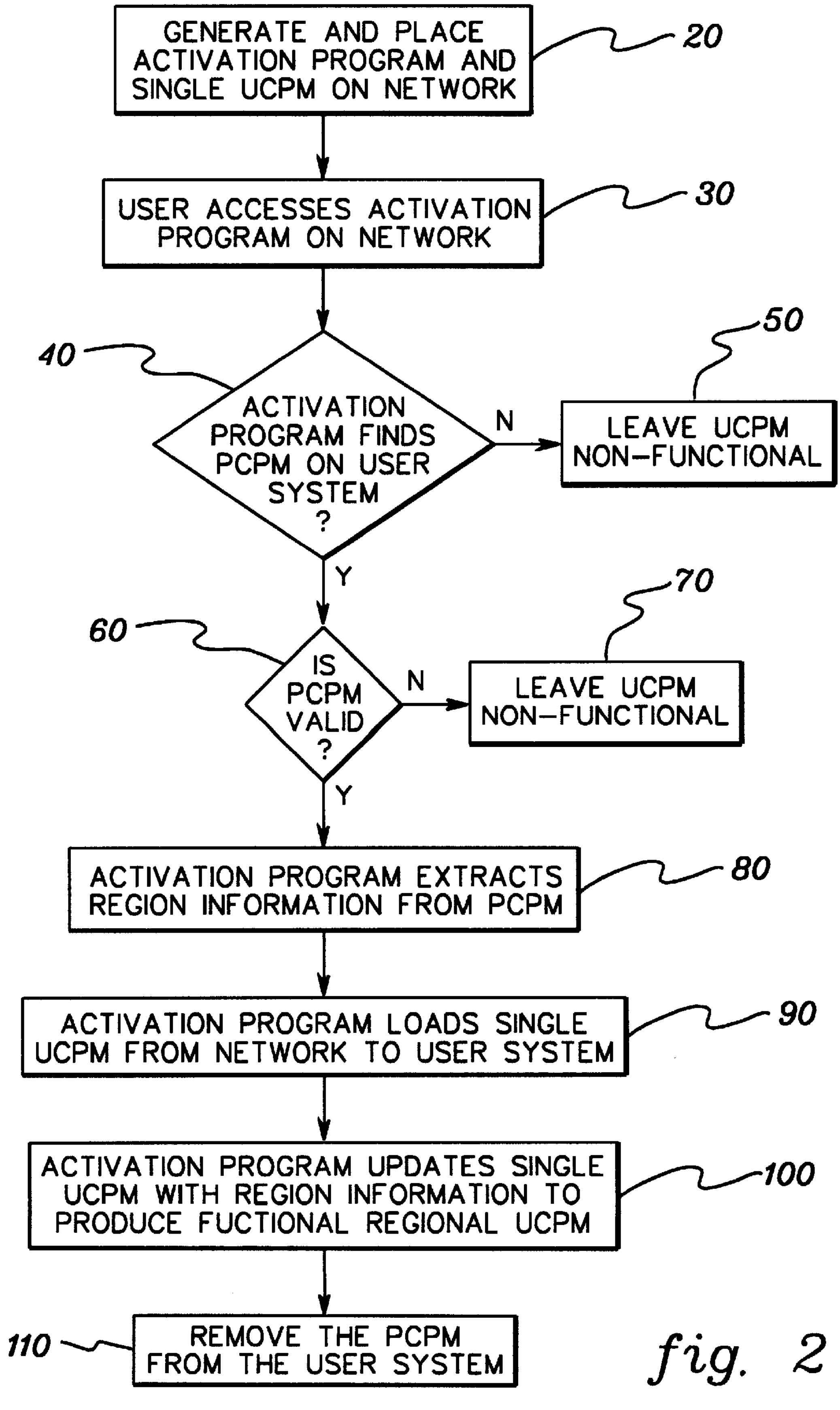
24 Claims, 2 Drawing Sheets





LOCAL DVD USER SYSTEM – REGION 1
LOCAL DVD USER SYSTEM – REGION 2
LOCAL DVD USER SYSTEM – REGION N
16
16
NETWORK
14
10
SINGLE NONFUNCTIONAL UCPM WITH ACTIVATION PROGRAM
OEM SYSTEM
12

fig. 1



GENERATE AND PLACE ACTIVATION PROGRAM AND SINGLE UCPM ON NETWORK
20
USER ACCESSES ACTIVATION PROGRAM ON NETWORK
30
40
ACTIVATION PROGRAM FINDS PCPM ON USER SYSTEM ?
N
LEAVE UCPM NON-FUNCTIONAL
50
Y
60
IS PCPM VALID ?
N
LEAVE UCPM NON-FUNCTIONAL
70
Y
ACTIVATION PROGRAM EXTRACTS REGION INFORMATION FROM PCPM
80
ACTIVATION PROGRAM LOADS SINGLE UCPM FROM NETWORK TO USER SYSTEM
90
ACTIVATION PROGRAM UPDATES SINGLE UCPM WITH REGION INFORMATION TO PRODUCE FUCTIONAL REGIONAL UCPM
100
110
REMOVE THE PCPM FROM THE USER SYSTEM

fig. 2

METHOD, SYSTEM AND COMPUTER PROGRAM PRODUCT FOR UPDATING REGION-DEPENDENT SOFTWARE USING A COMMON UPDATE MODULE FOR MULTIPLE REGIONS

TECHNICAL FIELD

The present invention relates in general to digital video disc technology and, more particularly, to a method, system and computer program product for updating region-dependent DVD software using a single update module placed onto a public network without compromising the unique region information incorporated within each DVD PC user system.

BACKGROUND OF THE INVENTION

Digital versatile disc (DVD) is an emerging technology which due to its nature, requires extensive encryption in order to protect the data, such as a motion picture, against unauthorized copying. DVD is a specification for the content of video, audio and other compressed data to be used as playback video, audio and, for example, subtitle data by a DVD decoder. The DVD video data is specified in the moving picture experts group (MPEG) standard (ISO/IEC 13818-2). As well as being represented by the standard, the data is also encrypted for protection.

Encryption is achieved using the industry's content scrambling system (CSS), which produces an encrypted, encoded data stream for DVD playback. The data stream can be decrypted, for example, by hardware licensed to perform CSS decryption. Conventionally, CSS decryption occurs at a PCI card, which also conventionally includes MPEG decompression of the encrypted, encoded data signal.

A DVD PC system or player is manufactured under license of the DVD consortium to include a copy protect module to protect distribution of DVD data worldwide. This technology includes scrambling of data and the concept of DVD regions throughout the world. Each digital video disc includes data that determines in which of the various defined regions of the world that DVD will play. Each DVD player or PC system sold in a given region must only play digital video discs that are allowed to be played in that region. Currently, the global market is divided into six DVD regions and each digital video disc produced contains data representative of regional information where the disc can be played.

In view of the region-dependent nature of DVD PC systems, there would be commercial advantage to a global technique to update the PC software of different DVD systems sold in the various DVD regions of the world without negatively effecting the region-dependent nature of the DVD systems.

DISCLOSURE OF THE INVENTION

This application addresses the problem of distribution of software updates, e.g., updates to the copy protection module, for DVD PC systems in various regions of the world. In particular, this application seeks to provide a single global update mechanism which can be used to update software modules which are region specific without affecting each DVD system's region specificity. Preferably, the single global update is distributed via a public network, such as the internet, to users in the multiple DVD regions.

Briefly described, the invention comprises in a first aspect, a method for updating pre-existing region-dependent

software. The method includes: extracting region information from the pre-existing region-dependent software located within a user system; loading a non-functional software update onto the user system, the update comprising a non-functional software update for multiple regions; and, merging the region information extracted from the pre-existing region-dependent software with the non-functional software update loaded onto the user system to produce functional, updated region-dependent software.

In another aspect, the invention comprises a system for updating pre-existing region-dependent software. This system includes means for extracting region information from the pre-existing region-dependent software located within a user system, and means for loading a non-functional software update onto the user system. The update comprises a common non-functional software update for multiple regions. The system also has means for merging the region information extracted from the pre-existing region-dependent software with the non-functional software update loaded onto the user system to produce functional, updated region-dependent software.

A computer program product implementing the technique of the above-summarized method and system is also described and claimed herein.

Advantageously, a method, system or computer program product in accordance with the principles of this invention allows any user in the world to download a common software package, such as an updated copy protection module, and a corresponding activation module, and use the same process (i.e., instructions) to activate the updated module, while still maintaining each user's unique region information. The users will have no input or ability to change the region information on their system, which is a DVD requirement.

BRIEF DESCRIPTION OF DRAWINGS

The above-described objects, advantages and features of the present invention, as well as others, will be more readily understood from the following detailed description of certain preferred embodiments of the invention, when considered in conjunction with the accompanying drawings in which:

FIG. **1** depicts one example of a computing environment incorporating and using the software update facility of the present invention; and

FIG. **2** is a flowchart of one embodiment of the software update facility of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

Generally stated, the present invention comprises a method, system and computer program product for updating pre-existing region-dependent software, such as the copy protect module employed within DVD PC systems. A DVD PC system comprises a software implementation of the hardware logic implemented within a DVD player.

As noted initially, the DVD consortium licenses technology which includes the concept of DVD geographical regions. PC original equipment manufacturers (OEM's) implementing DVD technology require six different copy protection modules, one for each of the six DVD regions existing today. These modules are equivalent with the exception of certain unique region information. It is the responsibility of the PC OEM to ship the correct software module associated with the destination region of that DVD PC

system. For example, if a DVD PC system is shipped to the United States, then a Region 1 software module must be provided. If the DVD PC system is shipped to Japan, then a Region 2 software module must be provided. The PC OEM provides customers with one, and only one, software module within a given region per requirement of the DVD consortium.

One software module of particular importance is the copy protection module (CPM). DVD technology dictates extensive copy protection. One aspect of this requirement is addressed for DVD PC systems in co-pending, commonly assigned U.S. patent application Ser. No. 08/881,139, by Ciacelli et al., entitled "Apparatus, Method and Computer Program Product For Protecting Copyright Data Within a Computer System," the entirety of which is hereby incorporated herein by reference. Briefly, this patent describes a facility for re-encrypting CSS descrambled data within the CPU prior to its transfer from the CPU to maintain integrity of the copyrighted material, while still allowing software descrambling of the CSS encrypted data stream. Various techniques for establishing the encryption/decryption algorithm pair are presented therein.

With the above as background, the problem addressed herein concerns the distribution of software updates, and in particular updates for the copy protection module. Preferably, distribution of software fixes is accomplished by placing an updated module on a public network, such as the internet, and allowing customers to access, download, and install the updated software. However, in the case of region-specific modules, placing all six module updates on the public network would allow anyone to download software for any or all regions. This would allow users to update their DVD software to support all regions, which defeats the ability of the DVD consortium to distribute digital video discs based on a global model of multiple regions. Thus, presented herein is a novel facility for a PC OEM to distribute fixes to DVD region-specific software via a public network to customers in multiple DVD regions without impairing the region-dependent nature of the DVD systems.

Assume that the PC OEM has shipped one and only one copy protection module (CPM) with each DVD system, and that the CPM is consistent with the DVD region as defined by the DVD consortium. In this case, the correct region information for that DVD system is contained within the CPM that exists (herein the "pre-existing copy protection module" (PCPM)) on the user's system. One embodiment of a computer environment **10** incorporating and using the update facility of the present invention is depicted in FIG. **1**. Environment **10** includes an OEM system **12** coupled to a network **14** (such as the internet) and multiple local DVD user systems **16** distributed within N regions. Any given DVD region may contain a large number of local DVD user systems. Each local DVD user system is assumed to comprise a DVD PC system, such as that described in the above-designated co-pending application.

Generally, the solution of the present invention is to employ a common "updated copy protection module" (UCPM) with the appropriate region information removed such that the UCPM is non-functional. This non-functional UCPM is then placed on the public network for distribution. To activate the module, an activation program or module is also made available with the UCPM, which has several characteristics.

The activation module will have tamper-resistant methodology applied to it. For example, reference the above-designated, incorporated patent application for various tamper resistant processing approaches. This will protect the information contained within the module. Operationally, the activation program first locates the PCPM installed on the user system. If a pre-existing module is not found, or has been tampered with, the network available module (UCPM) will not be activated on the user's system and will be of no value.

If the activation program finds the PCPM, then it will validate the integrity of the existing module. Based on several imbedded tags, and tables of valid modules, the activation program can determine if this pre-existing software is valid, which software module it is (i.e., assuming multiple versions or revision levels exist), and the location within the module of the region information. If valid, the activation module extracts the region information therefrom.

At this point, the activation module loads the updated copy protection module (UCPM) obtained from the public network, to an appropriate location on the user's hard disc. The activation module then updates the new "non-functional" module (UCPM) with the region information obtained from the pre-existing copy protection module (PCPM) that is being updated. The new program is then functional and supports the appropriate DVD region, and the old software module is removed from the system by the activation program.

FIG. **2** presents a flowchart overview of the software update facility in accordance with the present invention. As shown, the single updated copy protection module (UCPM) is generated and placed on the network, along with the activation module **20**. A remote user then accesses the activation program on the network **30**. The activation program locates the pre-existing copy protect module (PCPM) on the user's system **40**, e.g., by searching the user's hard drive for a known module name. If no PCPM is located, then the UCPM remains on the network and non-functional to the user **50**.

Assuming that the PCPM is located on the user system, the activation module determines whether the PCPM is valid **60**, e.g., by examining certain defined fields of the module. Again if no, the updated module remains on the network and non-functional to the user **70**. If the PCPM is valid, then the activation program extracts the regional information from a predefined field of the PCPM **80**. The activation program thereafter loads the single "non-functional" UCPM from the network onto the user's system **90**. Although loading of the UCPM from the network to the user can precede evaluation of the PCPM and in particular locating of the region information, it is believed preferable to await extraction of proper region information before loading the update onto the user's network. Thereafter, the activation module updates the single UCPM by merging the region information into an appropriate field of the updated code to produce a functional, updated copy protect module **100** for the user's region of use. Finally, the activation program removes the PCPM from the user's system **110**.

Those skilled in the art will note from the above discussion that presented herein is an approach which would allow any user in the world to download software, such as an updated copy protection module, and corresponding activation module, and use the same process (instructions) to activate the updated module, while still maintaining each user's unique region information. The users will have no input or ability to change the region information on their system, which is a DVD requirement.

The present invention can be included, for example, in an article of manufacture (e.g., one or more computer program

products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The articles manufactured can be included as part of the computer system or sold separately.

The flow diagrams depicted herein are provided by way of example. There may be variations to these diagrams or the steps or operations described herein without departing from the spirit of the invention. For instance, in certain cases the steps may be performed in differing order, or steps may be added, deleted or modified. All these variations are considered to comprise part of the present invention as recited in the appended claims.

While the invention has been described in detail herein in accordance with certain preferred embodiments thereof, many modifications and changes therein may be effected by those skilled in the art. Accordingly, it is intended by the appended claims to cover all such modifications and changes as fall within the true spirit and scope of the invention.

I claim:

1. A method for updating pre-existing region-dependent software, said method comprising:

extracting region information from the pre-existing region-dependent software located within a user system;

loading a non-functional software update onto the user system, said update comprising a common non-functional software update for multiple regions; and

merging the region information extracted from the pre-existing region-dependent software with the non-functional software update loaded onto the user system to produce functional, updated region-dependent software.

2. The method of claim **1**, wherein said pre-existing region-dependent software comprises a pre-existing copy protect module and said non-functional software update comprises an updated copy protect module.

3. The method of claim **1**, further comprising providing said non-functional software update to multiple regions.

4. The method of claim **3**, wherein said providing includes placing said common non-functional software update on a network, and wherein multiple user systems are coupled to said network within said multiple regions.

5. The method of claim **4**, wherein said providing further comprises providing an activation module to said multiple regions in association with said non-functional software update, said activation module accomplishing said extracting, said loading, and said merging.

6. The method of claim **1**, further comprising prior to said extracting, verifying presence of said pre-existing region-dependent software on said user system.

7. The method of claim **6**, further comprising prior to said extracting, determining whether the pre-existing region-dependent software on said user system is valid, and if invalid, discontinuing said extracting, said loading and said merging.

8. The method of claim **1**, further comprising removing the pre-existing region-dependent software from the user system subsequent to producing said functional, updated region-dependent software.

9. The method of claim **1**, wherein said user system comprises a digital video disc (DVD) PC system, and wherein said loading comprises loading said non-functional software update onto the DVD PC system.

10. A system for updating pre-existing region-dependent software, said system comprising:

means for extracting region information from the pre-existing region-dependent software located within a user system;

means for loading a non-functional software update onto the user system, said update comprising a common non-functional software update for multiple regions; and

means for merging the region information extracted from the pre-existing region-dependent software with the non-functional software update loaded onto the user system to produce functional, updated region-dependent software.

11. The system of claim **10**, wherein said pre-existing region-dependent software comprises a pre-existing copy protect module and said non-functional software update comprises an updated copy protect module.

12. The system of claim **10**, further comprising means for providing said non-functional software update to multiple regions, said means for providing comprising means for placing said non-functional software update on a network, and wherein multiple user systems are coupled to said network within said multiple regions.

13. The system of claim **12**, wherein said means for providing further comprises means for providing an activation module to said multiple regions in association with said non-functional software update, said activation module comprising said means for extracting, means for loading, and means for merging.

14. The system of claim **10**, further comprising means for verifying presence of said pre-existing region-dependent software on said user system prior to extracting of said regional information.

15. The system of claim **14**, further comprising means for determining prior to said extracting whether the pre-existing region-dependent software on said system is valid, and if invalid, means for discontinuing processing of said means for extracting, means for loading and means for merging.

16. The system of claim **10**, further comprising means for removing the pre-existing region-dependent software from the user system subsequent to producing said functional, updated region-dependent software.

17. The system of claim **10**, wherein said user system comprises a digital video disc (DVD) PC system.

18. A computer program product comprising a computer usable medium having computer readable program code means therein for use in updating pre-existing region-dependent software, said computer readable program code means in said computer program product comprising:

computer readable program code means for causing a computer to affect extracting region information from the pre-existing region-dependent software located within a user system;

computer readable program code means for causing a computer to affect loading a non-functional software update onto the user system, said update comprising a common non-functional software update for multiple regions; and

computer readable program code means for causing a computer to affect merging the region information extracted from the pre-existing region-dependent software with the non-functional software update loaded onto the user system to produce functional, updated region-dependent software.

19. The computer readable program code means of claim **18**, wherein said pre-existing region-dependent software comprises a pre-existing copy protect module and said

non-functional software update comprises an updated copy protect module.

20. The computer readable program code means of claim **18**, further comprising computer readable program code means for causing a computer to affect providing said non-functional software update to multiple regions across a network, and wherein multiple user systems are coupled to said network within said multiple regions.

21. The computer readable program code means of claim **20**, wherein said computer readable program code means for causing a computer to affect providing further comprises computer readable program code means for causing a computer to affect providing an activation module to said multiple regions in association with said non-functional software update, said activation module comprising said computer readable program code means for causing a computer to affect said extracting, said loading and said merging.

22. The computer readable program code means of claim **18**, further comprising computer readable program code means for verifying presence of said pre-existing region-dependent software on said user system prior to extracting of said regional information.

23. The computer readable program code means of claim **22**, further comprising computer readable program code means for causing a computer to affect determining whether the pre-existing region-dependent software on said user system is valid, and if invalid, for discontinuing said extracting, said loading and said merging.

24. The computer readable program code means of claim **18**, wherein said user system comprises a digital video disc (DVD) PC system.

* * * * *

US006097757A

United States Patent [19]

Boice et al.

[11] Patent Number: 6,097,757

[45] Date of Patent: Aug. 1, 2000

[54] **REAL-TIME VARIABLE BIT RATE ENCODING OF VIDEO SEQUENCE EMPLOYING STATISTICS**

[75] Inventors: **Charles E. Boice**, Endicott, N.Y.; **Adrian S. Butter**, Sunnyvale, Calif.; **Agnes Y. Ngai**, Endwell, N.Y.; **Nader Mohsenian**, Endicott, N.Y.; **Robert Woodard**, Newark Valley, N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **09/008,282**

[22] Filed: **Jan. 16, 1998**

[51] **Int. Cl.**[7] **H04N 7/36**; H04N 7/50

[52] **U.S. Cl.** **375/240**; 348/415

[58] **Field of Search** 375/240; 348/415, 348/409, 416, 417

[56] **References Cited**

U.S. PATENT DOCUMENTS

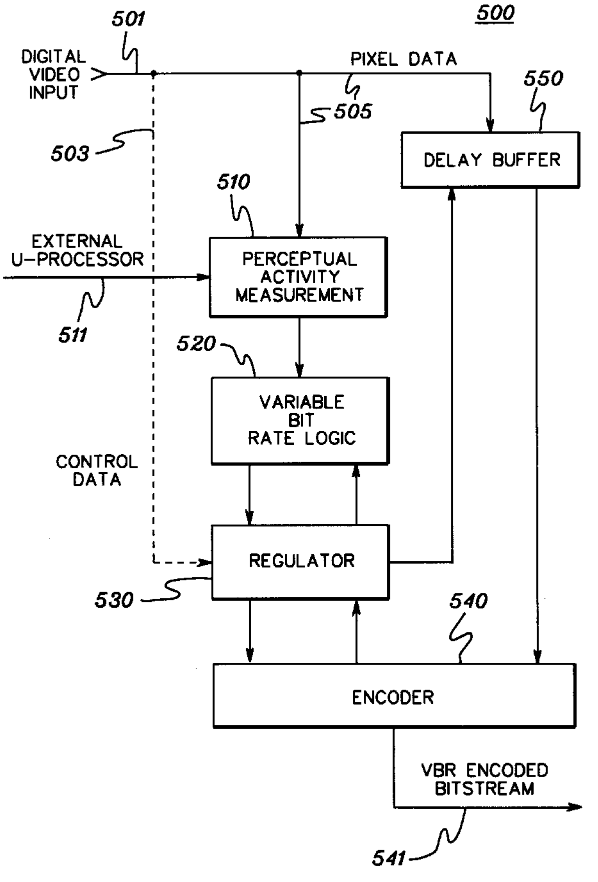
5,287,182	2/1994	Haskell et al.	348/500
5,301,242	4/1994	Gonzales et al.	382/56
5,365,272	11/1994	Siracusa	348/426
5,440,346	8/1995	Alattar et al.	348/420
5,469,212	11/1995	Lee	348/392
5,483,287	1/1996	Siracusa	348/426
5,781,788	7/1998	Woo et al.	712/1
5,793,895	8/1998	Chang et al.	382/236
5,978,029	11/1999	Boice et al.	348/412
6,040,861	3/2000	Boroczky et al.	348/409

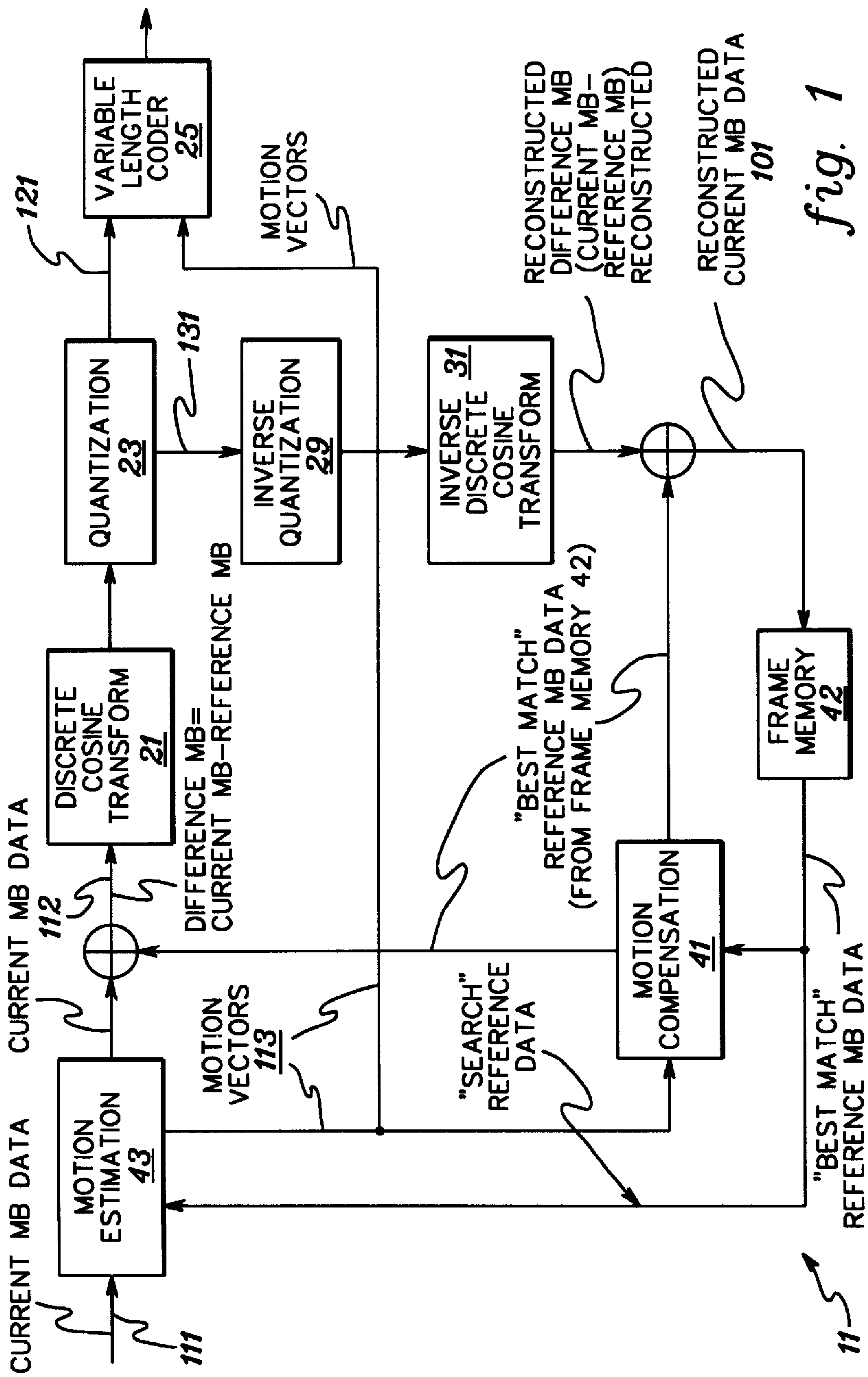
Primary Examiner—Howard Britton
Attorney, Agent, or Firm—William H. Steinberg, Esq.; Heslin & Rothenberg, P.C.

[57] **ABSTRACT**

Method, system and computer program product are provided for adaptively encoding in hardware, software or a combination thereof a sequence of video frames in real-time. Pre-encode perceptual activity measurement processing is employed to derive statistics on each frame of the sequence of video frames to be encoded. The statistics are used by variable bit rate logic to obtain a number of bits to be used in encoding each frame. The number of bits to be used is provided to a single encoding engine, which encodes the sequence of video frames and produces a constant quality, variable bit rate bitstream output. The pre-encode processing employs a regulator as the global data flow control and synchronization for the encoder. Perceptual activity analysis on each frame of the sequence of video frames can derive information on, for example, shading, scene change, fade, color, motion and/or edge presence within the frame. Voting gives greater weight to the presence of certain characteristics within the frame.

30 Claims, 8 Drawing Sheets





CURRENT MB DATA
111
MOTION ESTIMATION 43
CURRENT MB DATA
112
DISCRETE COSINE TRANSFORM 21
DIFFERENCE MB= CURRENT MB–REFERENCE MB
QUANTIZATION 23
121
VARIABLE LENGTH CODER 25
131
INVERSE QUANTIZATION 29
MOTION VECTORS
MOTION VECTORS 113
"SEARCH" REFERENCE DATA
"BEST MATCH" REFERENCE MB DATA (FROM FRAME MEMORY 42)
INVERSE DISCRETE COSINE TRANSFORM 31
RECONSTRUCTED DIFFERENCE MB (CURRENT MB– REFERENCE MB) RECONSTRUCTED
RECONSTRUCTED CURRENT MB DATA 101
MOTION COMPENSATION 41
FRAME MEMORY 42
"BEST MATCH" REFERENCE MB DATA
11

fig. 1

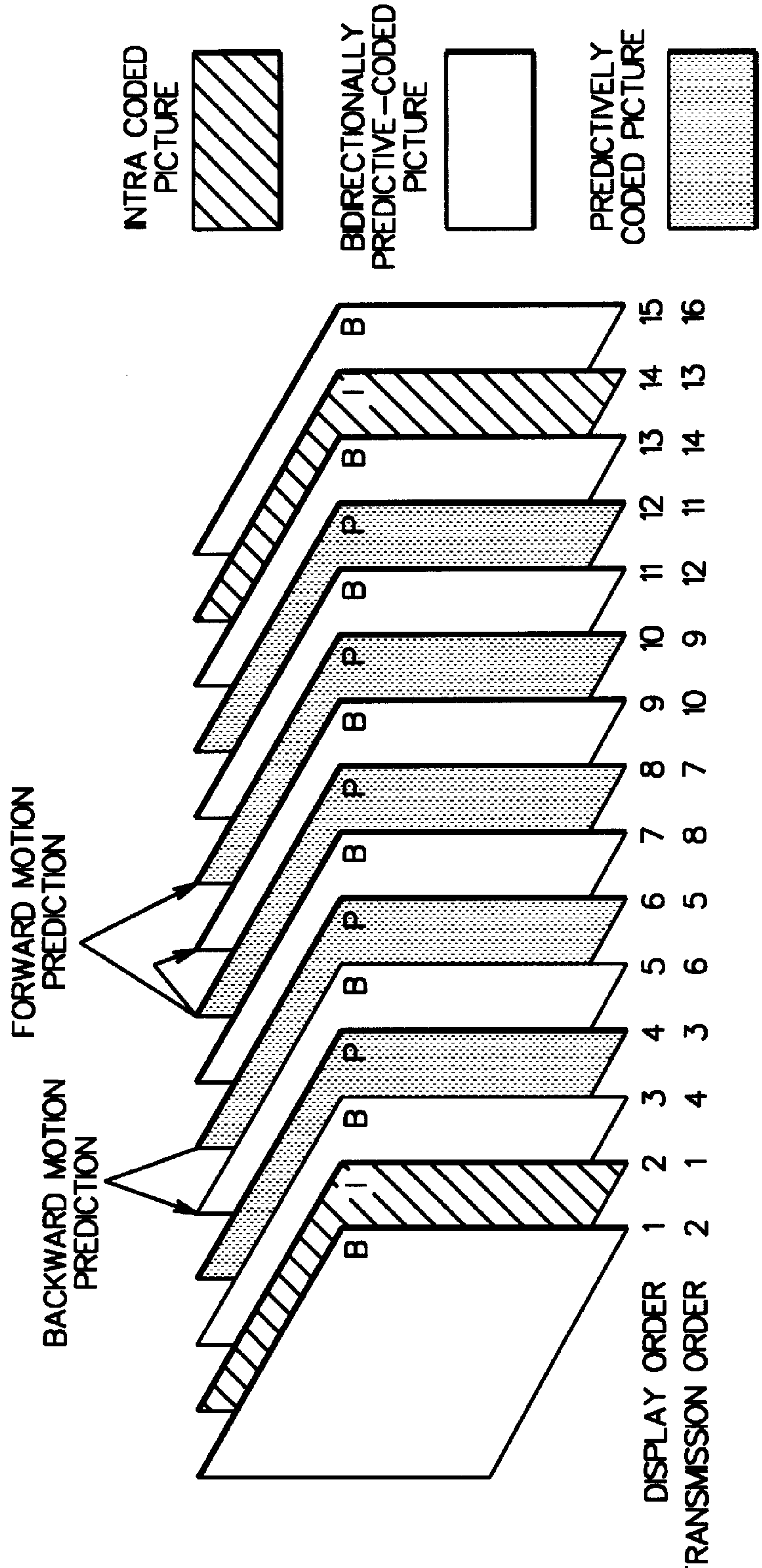

INTRA CODED PICTURE
BIDIRECTIONALLY PREDICTIVE–CODED PICTURE
PREDICTIVELY CODED PICTURE
FORWARD MOTION PREDICTION
BACKWARD MOTION PREDICTION
B I B P B P B P B P B P B I B
DISPLAY ORDER 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
TRANSMISSION ORDER 2 1 4 3 6 5 8 7 10 9 12 11 14 13 16

fig. 2

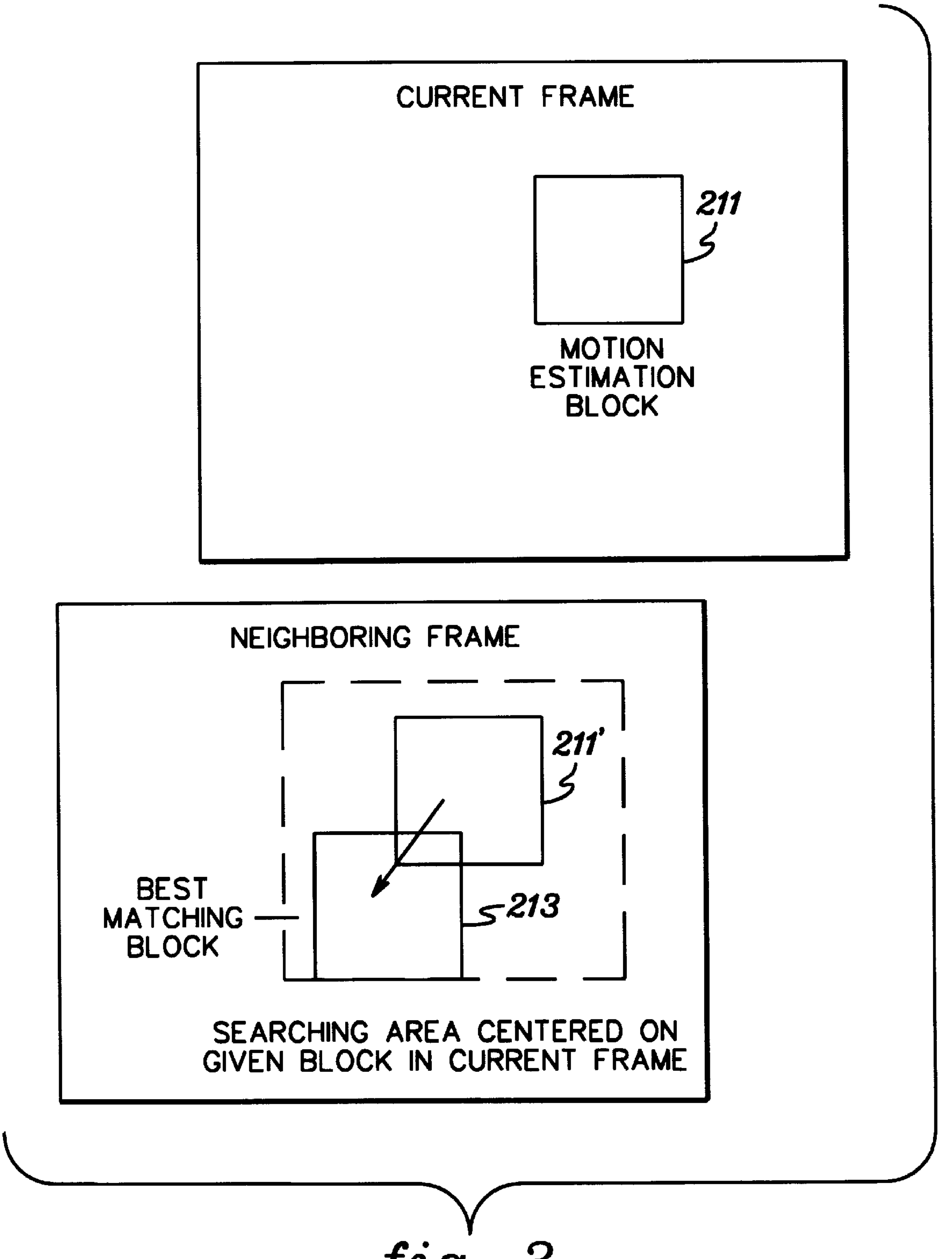

CURRENT FRAME
211
MOTION ESTIMATION BLOCK
NEIGHBORING FRAME
211'
213
BEST MATCHING BLOCK
SEARCHING AREA CENTERED ON GIVEN BLOCK IN CURRENT FRAME

fig. 3

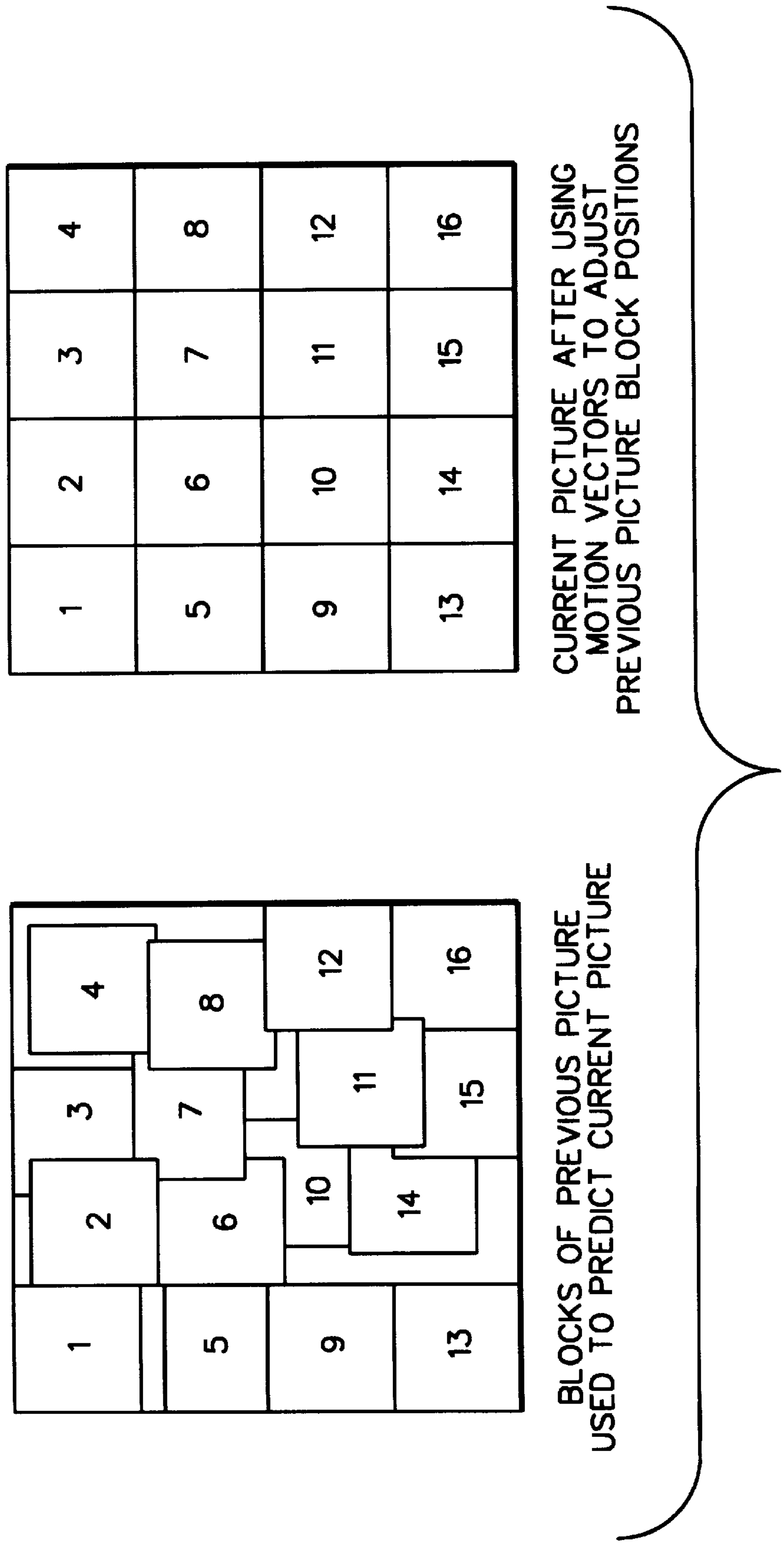

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
CURRENT PICTURE AFTER USING MOTION VECTORS TO ADJUST PREVIOUS PICTURE BLOCK POSITIONS
BLOCKS OF PREVIOUS PICTURE USED TO PREDICT CURRENT PICTURE

fig. 4

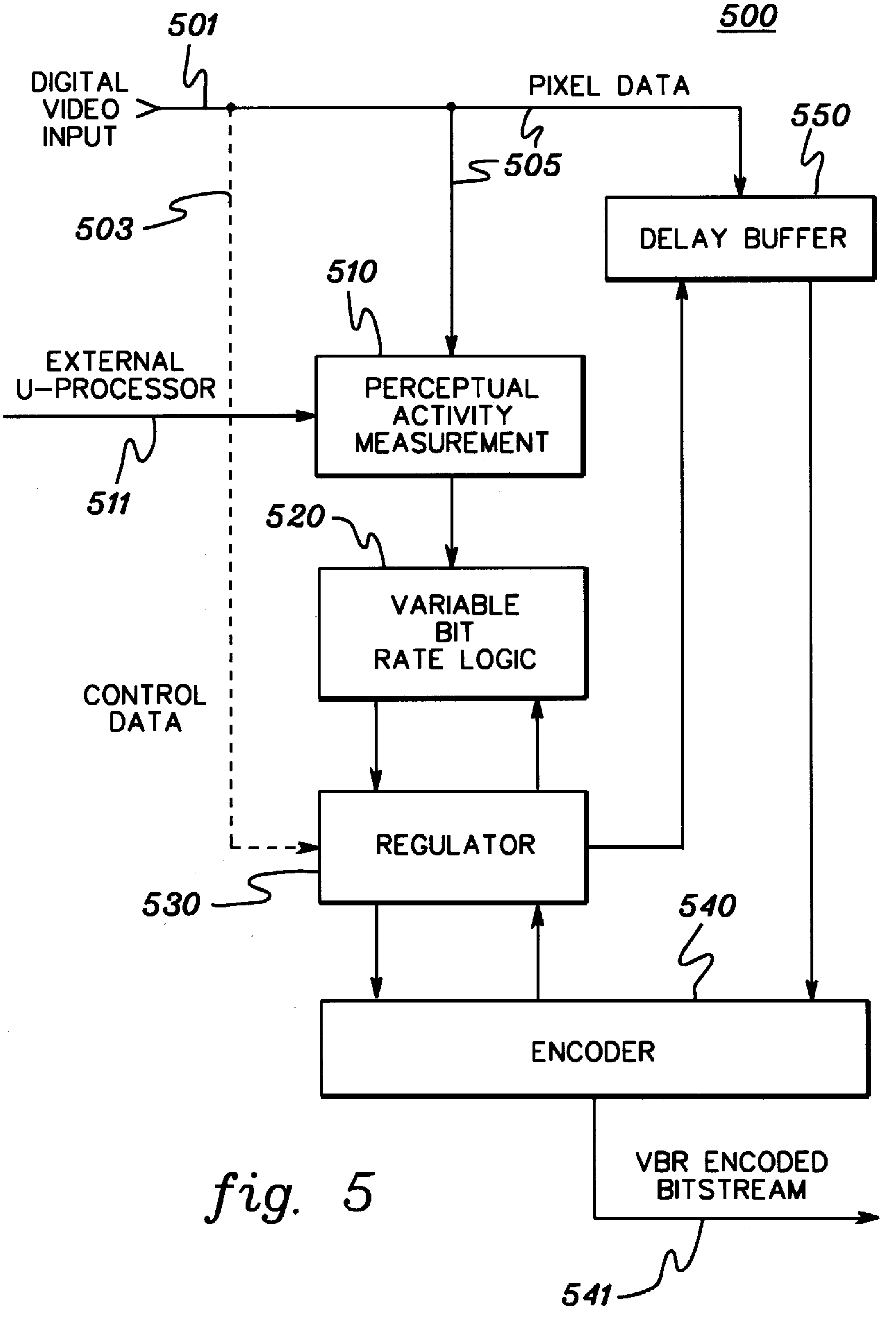

500
501
DIGITAL VIDEO INPUT
PIXEL DATA
505
550
DELAY BUFFER
503
510
EXTERNAL U-PROCESSOR
PERCEPTUAL ACTIVITY MEASUREMENT
511
520
VARIABLE BIT RATE LOGIC
CONTROL DATA
REGULATOR
530
540
ENCODER
VBR ENCODED BITSTREAM
541

fig. 5

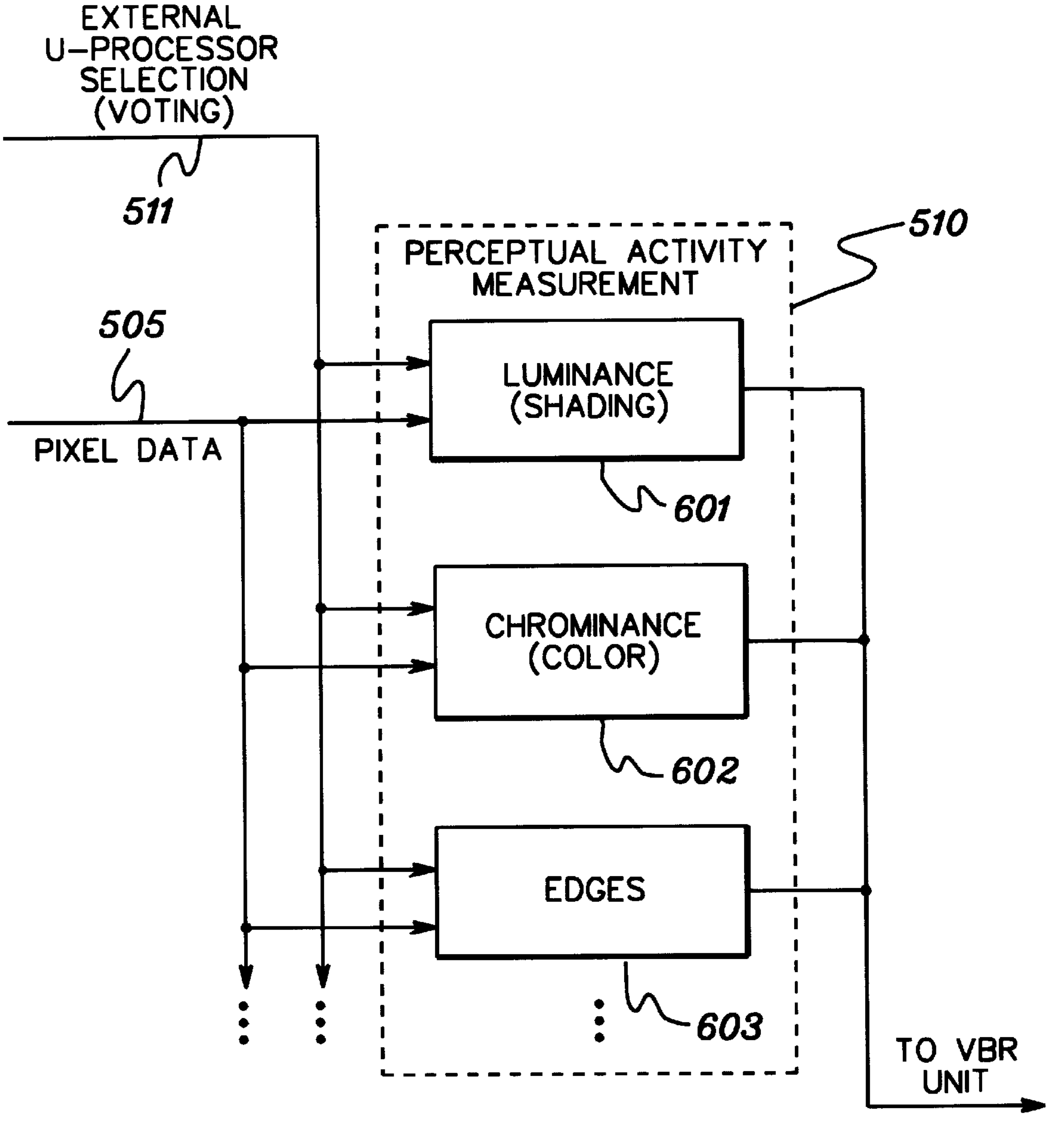
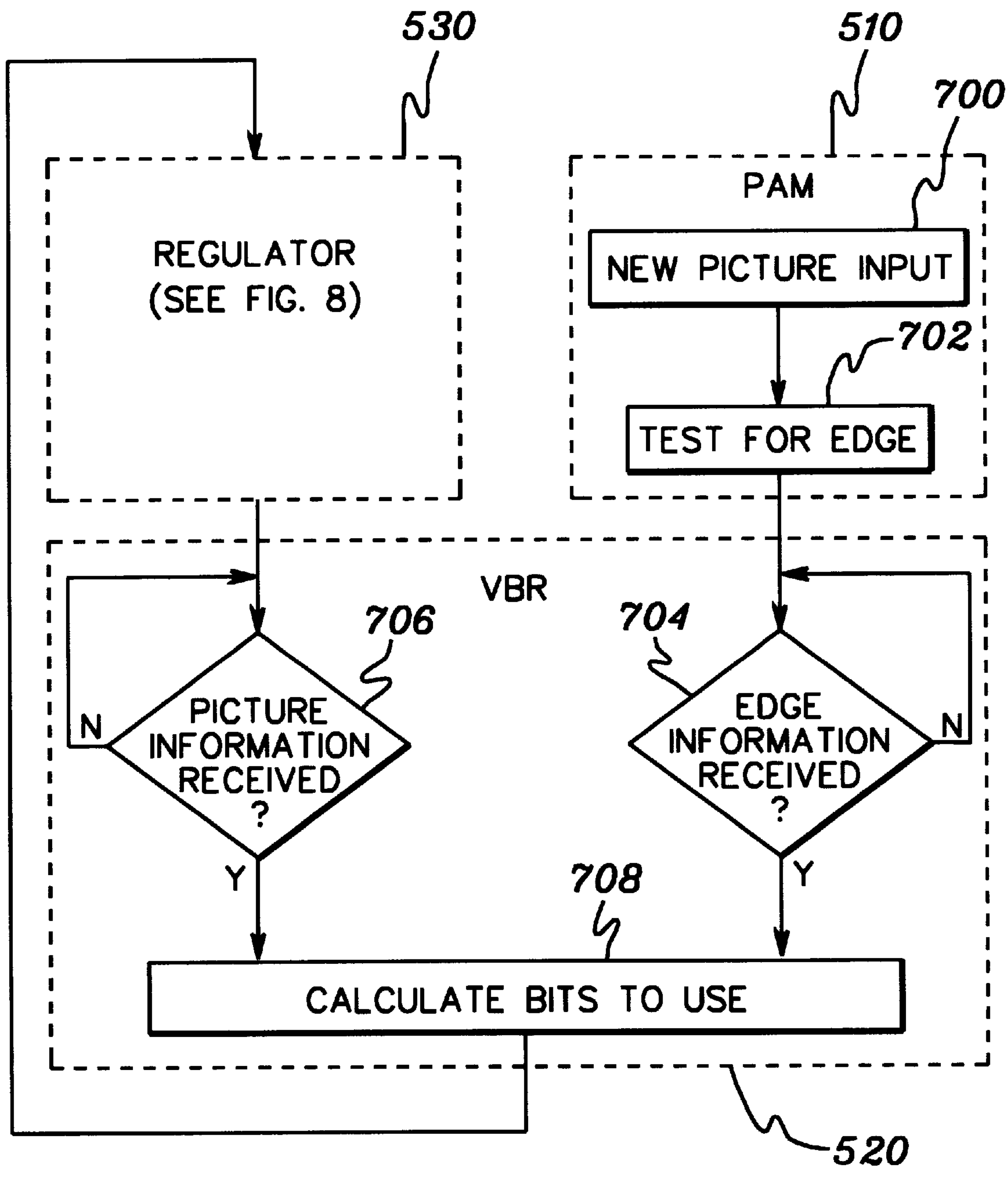
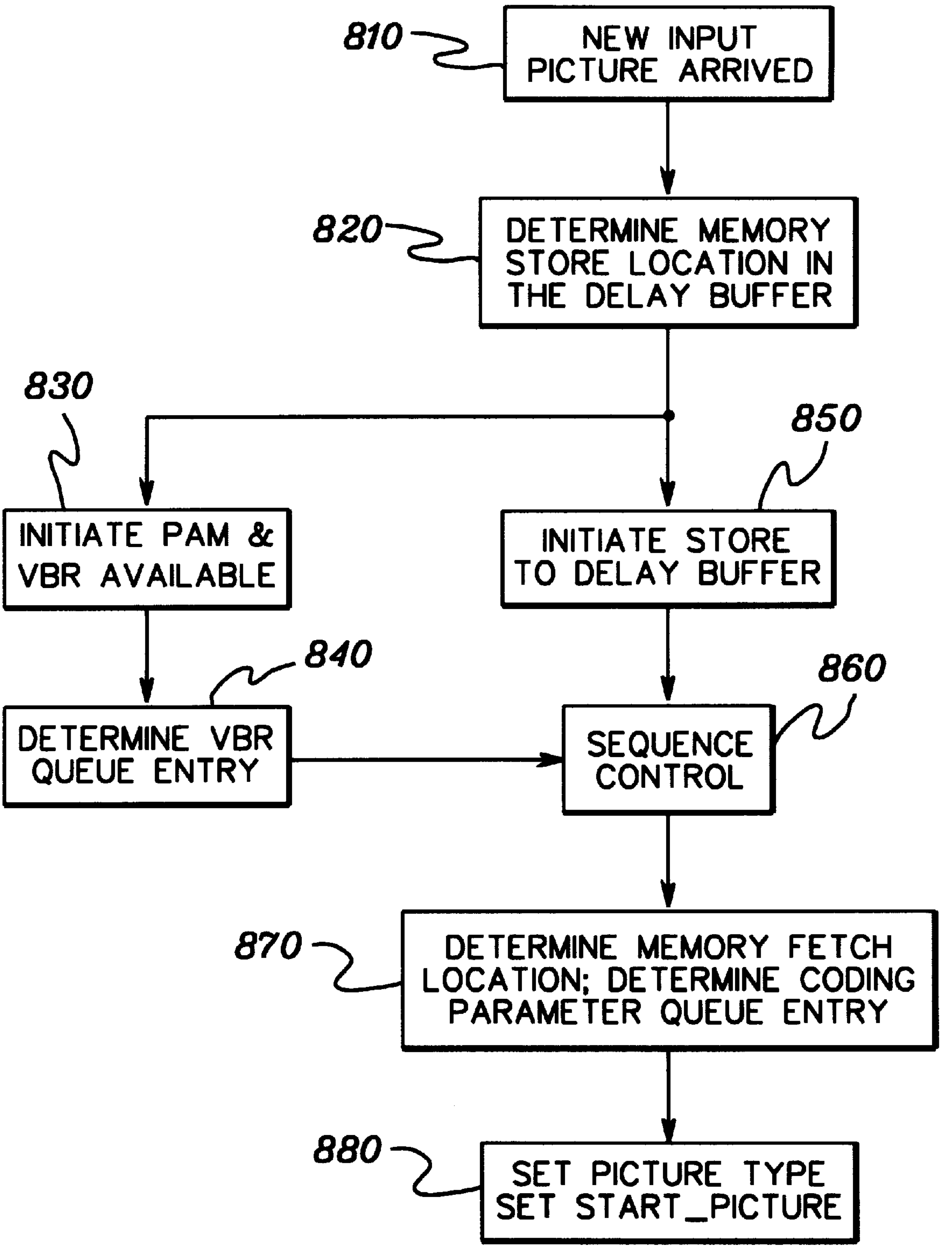

EXTERNAL U-PROCESSOR SELECTION (VOTING)
511
510
PERCEPTUAL ACTIVITY MEASUREMENT
505
LUMINANCE (SHADING)
PIXEL DATA
601
CHROMINANCE (COLOR)
602
EDGES
603
TO VBR UNIT

fig. 6



530
510
700
PAM
REGULATOR
(SEE FIG. 8)
NEW PICTURE INPUT
702
TEST FOR EDGE
VBR
706
704
PICTURE INFORMATION RECEIVED ?
EDGE INFORMATION RECEIVED ?
N
N
Y
Y
708
CALCULATE BITS TO USE
520

fig. 7



810
NEW INPUT PICTURE ARRIVED
820
DETERMINE MEMORY STORE LOCATION IN THE DELAY BUFFER
830
INITIATE PAM & VBR AVAILABLE
850
INITIATE STORE TO DELAY BUFFER
840
DETERMINE VBR QUEUE ENTRY
860
SEQUENCE CONTROL
870
DETERMINE MEMORY FETCH LOCATION; DETERMINE CODING PARAMETER QUEUE ENTRY
880
SET PICTURE TYPE SET START_PICTURE

fig. 8

REAL-TIME VARIABLE BIT RATE ENCODING OF VIDEO SEQUENCE EMPLOYING STATISTICS

CROSS-REFERENCE TO RELATED APPLICATION

This application is related to commonly assigned, United States Patent Application by Boroczky et al., entitled "Adaptive Real-Time Encoding Of Video Sequence Employing Image Statistics," filed Oct. 10, 1997, Ser. No. 08/948,442, now U.S. Pat. No. 6,040,861, the entirety of which is hereby incorporated herein by reference.

TECHNICAL FIELD

The present invention relates in general to compression of digital video images, and more particularly, to a technique for using image statistics derived from a video sequence to dynamically change one or more controllable encoding parameter(s) from frame to frame or within a frame to achieve real-time, variable bit rate encoding of the video sequence.

BACKGROUND OF THE INVENTION

Within the past decade, the advent of world-wide electronic communications systems has enhanced the way in which people can send and receive information. In particular, the capabilities of real-time video and audio systems have greatly improved in recent years. In order to provide services such as video-on-demand and video conferencing to subscribers, an enormous amount of network bandwidth is required. In fact, network bandwidth is often the main inhibitor in the effectiveness of such systems.

In order to overcome the constraints imposed by networks, compression systems have emerged. These systems reduce the amount of video and audio data which must be transmitted by removing redundancy in the picture sequence. At the receiving end, the picture sequence is uncompressed and may be displayed in real-time.

One example of an emerging video compression standard is the Moving Picture Experts Group ("MPEG") standard. Within the MPEG standard, video compression is defined both within a given picture and between pictures. Video compression within a picture is accomplished by conversion of the digital image from the time domain to the frequency domain by a discrete cosine transform, quantization, and variable length coding. Video compression between pictures is accomplished via a process referred to as motion estimation and compensation, in which a motion vector plus difference data is used to describe the translation of a set of picture elements (pels) from one picture to another.

The ISO MPEG-2 standard specifies only the syntax of bitstream and semantics of the decoding process. The choice of coding parameters and tradeoffs in performance versus complexity are left to the encoder developers.

In video applications, it is advantageous to optimize encoding of digital signals in order to obtain the best density or compression of data. There are a number of known techniques to accomplish encoding, but complex pictures with frequent inter-frame changes may require more time for compression than is available on a per-frame basis.

This invention therefore seeks to enhance picture quality of an encoded video sequence while still obtaining a high compression rate by providing a real-time, variable bit rate encoding scheme.

DISCLOSURE OF THE INVENTION

Briefly described, the invention comprises in one aspect a method for encoding a sequence of video frames using a single encoding engine. The method includes: performing perceptual activity analysis on a frame of the sequence of video frames to derive information on at least one characteristic thereof; setting at least one controllable parameter for use in encoding the frame of the sequence of video frames based upon the information derived on the at least one characteristic thereof; and, encoding the frame of the sequence of video frames in real-time using the single encoding engine and the at least one controllable parameter.

In another aspect, the invention comprises a system for encoding a sequence of video frames which includes a pre-encode processing unit and a single encoding engine. The pre-encode processing unit comprises a perceptual activity measurement unit and a variable bit rate unit. The perceptual activity measurement unit performs perceptual activity analysis on a frame of the sequence of video frames to derive information on at least one characteristic thereof. The variable bit rate unit sets at least one control parameter based upon the information derived on the at least one characteristic. The at least one controllable parameter is subsequently used by the single encoding engine in encoding the frame of the sequence of video frames in real-time.

In still another aspect, the invention comprises a computer program product having computer usable medium with computer readable program code means therein for use in encoding a sequence of video frames using a single encoding engine. The computer readable program code means in the computer program product includes computer readable program code means for causing a computer to affect: performing perceptual activity analysis on a frame of the sequence of video frames to derive information on at least one characteristic thereof; setting at least one controllable parameter for use in encoding the frame of the sequence of the video frames based upon the information derived on the at least one characteristic thereof; and, encoding the frame of the sequence of video frames in real-time using the single encoding engine and the at least one controllable parameter.

In general, encoding in accordance with the principles of the present invention results in improved picture quality compared with non-adaptive encoder systems, especially at low bit rates. This because, for example, employing adaptive bit allocation among frames (as well as within frames) is more critical in low bit rate encoding compared with higher bit rate encoding. Further, the encoding technique of this invention can insure a semi-constant picture quality of a decoded video sequence in constant bit rate (CBR) mode or a constant picture quality in variable bit rate (VBR) encoding mode.

BRIEF DESCRIPTION OF THE DRAWINGS

The above-described objects, advantages and features of the present invention, as well as others, will be more readily understood from the following detailed description of certain preferred embodiments of the invention, when considered in conjunction with the accompanying drawings in which:

FIG. **1** shows a flow diagram of a generalized MPEG-2 compliant encoder **11**, including a discrete cosine transformer **21**, a quantizer **23**, a variable length coder **25**, an inverse quantizer **29**, an inverse discrete cosine transformer **31**, motion compensation **41**, frame memory **42**, and motion estimation **43**. The data paths include the ith picture input **111**, difference data **112**, motion vectors **113** (to motion compensation **41** and to variable length coder **25**), the picture output **121**, the feedback picture for motion estimation and compensation **131**, and the motion compensated

picture **101**. This figure has the assumptions that the ith picture exists in frame memory or frame store **42** and that the i+1th picture is being encoded with motion estimation.

FIG. **2** illustrates the I, P, and B pictures, examples of their display and transmission orders, and forward, and backward motion prediction.

FIG. **3** illustrates the search from the motion estimation block in the current frame or picture to the best matching block in a subsequent or previous frame or picture. Elements **211** and **211**' represent the same location in both pictures.

FIG. **4** illustrates the movement of blocks in accordance with the motion vectors from their position in a previous picture to a new picture, and the previous picture's blocks adjusted after using motion vectors.

FIG. **5** shows one embodiment of a real-time encode system in accordance with the present invention. System **500** includes pre-encode processing comprising perceptual activity measurement unit **510**, variable bit rate logic **520** and regulator **530** which together dynamically derive information on at least one characteristic of a frame of a video sequence to be encoded. This information is employed by encoder **540** to adaptively change the encoding rate of the sequence of frames, thereby optimizing picture quality and/or encoding performance.

FIG. **6** is a generalized diagram of one embodiment of the perceptual activity measurement unit **510** of the real-time encode system **500** of FIG. **5**. Depicted by way of example, perceptual activity measurement **510** includes logic for evaluating luminance **601**, chrominance **602** and edges **603**, as well as other characteristics of the video sequence.

FIG. **7** is a flowchart of one embodiment of pre-encode processing implemented by perceptual activity measurement (PAM) unit **510**, variable bit rate logic **520** and regulator **530** of the real-time encode system **500** of FIG. **5**.

FIG. **8** is a flowchart of one embodiment of processing implemented by regulator **530** of the real-time encode system of FIG. **5**.

BEST MODE FOR CARRYING OUT THE INVENTION

The invention relates, for example, to MPEG compliant encoders and encoding processes as described in "Information Technology-Generic coding of moving pictures and associated audio information: Video," Recommendation ITU-T H.262, ISO/IEC 13818-2, Draft International Standard, 1994. The encoding functions performed by the encoder include data input, spatial compression, motion estimation/compensation, macroblock type generation, data reconstruction, entropy coding, and data output. Spatial compression includes discrete cosine transformation (DCT), quantization, and entropy encoding. Temporal compression includes intensive reconstructive processing, such as inverse discrete cosine transformation, inverse quantization, and motion compensation. Motion estimation and compensation are used for temporal compression functions. Spatial and temporal compression are repetitive functions with high computational requirements.

More particularly the invention relates, for example, to a process for performing spatial and temporal compression including discrete cosine transformation, quantization, entropy encoding, motion estimation, motion compensation, and prediction, and even more particularly to a system for accomplishing spatial and temporal compression.

The first compression step is the elimination of spatial redundancy, for example, the elimination of spatial redundancy in a still picture of an "I" frame picture. Spatial redundancy is the redundancy within a picture. The MPEG-2 Draft Standard is using a block based method of reducing spatial redundancy. The method of choice is the discrete cosine transformation, and discrete cosine transform coding of the picture. Discrete cosine transform coding is combined with weighted scalar quantization and run length coding to achieve efficient compression.

The discrete cosine transformation is an orthogonal transformation. Orthogonal transformations, because they have a frequency domain interpretation, are filter bank oriented. The discrete cosine transformation is also localized. That is, the encoding process samples on an 8×8 spatial window which is sufficient to compute 64 transform coefficients or sub-bands.

Another advantage of the discrete cosine transformation is that fast encoding and decoding algorithms are available. Additionally, the sub-band decomposition of the discrete cosine transformation is sufficiently well behaved to allow effective use of psychovisual criteria.

After transformation, many of the frequency coefficients are zero, especially the coefficients for high spatial frequencies. These coefficients are quantized and organized into a zig-zag or alternatescanned pattern, and converted into run-amplitude (run-level) pairs. Each pair indicates the number of zero coefficients and the amplitude of the non-zero coefficient. This is coded in a variable length code.

Motion compensation is used to reduce or even eliminate redundancy between pictures. Motion compensation exploits temporal redundancy by dividing the current picture into blocks, for example, macroblocks, and then searching in previously transmitted pictures for a nearby block with similar content. Only the difference between the current block pels and the predicted block pels extracted from the reference picture is actually compressed for transmission and thereafter transmitted.

The simplest method of motion compensation and prediction is to record the luminance and chrominance, i.e., intensity and color, of every pixel in an "I" picture, then record changes of luminance and chrominance, i.e., intensity and color for every specific pixel in the subsequent picture. However, this is uneconomical in transmission medium bandwidth, memory, processor capacity, and processing time because objects move between pictures, that is, pixel contents move from one location in one picture to a different location in a subsequent picture. A more advanced idea is to use a previous or subsequent picture to predict where a block of pixels will be in a subsequent or previous picture or pictures, for example, with motion vectors, and to write the result as "predicted pictures" or "P" pictures, or alternatively, as "bidirectionally predictive-coded pictures" or "B" pictures. More particularly, this involves making a best estimate or prediction of where the pixels or macroblocks of pixels of the ith picture will be in the i-1th or i+1th picture. It is one step further to use both subsequent and previous pictures to predict where a block of pixels will be in an intermediate or "B" picture.

To be noted is that the picture encoding order and the picture transmission order do not necessarily match the picture display order. See FIG. **2**. For I-P-B systems the input picture order is different from the encoding order, and the input pictures must be temporarily stored until used for encoding. A buffer stores this input until it is used.

For purposes of illustration, a generalized flowchart of MPEG compliant encoding is shown in FIG. **1**. In the flowchart the images of the ith picture and the i+1th picture

are processed to generate motion vectors. The motion vectors predict where a macroblock of pixels will be in a prior and/or subsequent picture. The use of the motion vectors is a key aspect of temporal compression in the MPEG standard. As shown in FIG. **1** the motion vectors, generated, are used for the translation of the macroblocks of pixels, from the i^{th} picture to the $i+1^{th}$ picture.

As shown in FIG. **1**, in the encoding process, the images of the i^{th} picture and the $i+1^{th}$ picture are processed in the encoder **11** to generate motion vectors which are the form in which, for example, the $i+1^{th}$ and subsequent pictures are encoded and transmitted. An input image **111** of a subsequent picture goes to the motion estimation unit **43** of the encoder. Motion vectors **113** are formed as the output of the motion estimation unit **43**. These vectors are used by the motion compensation unit **41** to retrieve macroblock data from previous and/or future pictures, referred to as "reference" data, for output by this unit. One output of the motion compensation unit **41** is negatively summed with the output from the motion estimation unit **43** and goes to the input of the discrete cosine transformer **21**. The output of the discrete cosine transformer **21** is quantized in a quantizer **23**. The output of the quantizer **23** is split into two outputs, **121** and **131**; one output **121** goes to a downstream element **25** for further compression and processing before transmission, such as to a run length encoder; the other output **131** goes through reconstruction of the encoded macroblock of pixels for storage in frame memory **42**. In the encoder shown for purposes of illustration, this second output **131** goes through an inverse quantization **29** and an inverse discrete cosine transform **31** to return a lossy version of the difference macroblock. This data is summed with the output of the motion compensation unit **41** and returns a lossy version of the original picture to the frame memory **42**.

As shown in FIG. **2**, there are three types of pictures. There are "Intra pictures" or "I" pictures which are encoded and transmitted, and do not require motion vectors to be defined. These "I" pictures serve as a reference image for motion estimation. There are "Predicted pictures" or "P" pictures which are formed by motion vectors from a previous picture and can serve as a reference image for motion estimation for further pictures. Finally, there are "Bidirectional pictures" or "B" pictures which are formed using motion vectors from two other pictures, one past and one future, and can not serve as a reference image for motion estimation. Motion vectors are generated from "I" and "P" pictures, and are used to form "P" and "B" pictures.

One method by which motion estimation is carried out, shown in FIG. **3**, is by a search from a macroblock **211** of an i^{th} picture throughout a region of the next picture to find the best match macroblock **213**. Translating the macroblocks in this way yields a pattern of macroblocks for the $i+1^{th}$ picture, as shown in FIG. **4**. In this way the i^{th} picture is changed a small amount, e.g., by motion vectors and difference data, to generate the $i+1^{th}$ picture. What is encoded are the motion vectors and difference data, and not the $i+1^{th}$ picture itself. Motion vectors translate position of an image from picture to picture, while difference data carries changes in chrominance, luminance, and saturation, that is, changes in shading and illumination.

Returning to FIG. **3**, we look for a good match by starting from the same location in the i^{th} picture as in the $i+1^{th}$ picture. A search window is created in the i^{th} picture. We search for a best match within this search window. Once found, the best match motion vectors for the macroblock are coded. The coding of the best match macroblock includes a motion vector, that is, how many pixels in the y direction and how many pixels in the x direction is the best match displaced in the next picture. Also encoded is difference data, also referred to as the "prediction error", which is the difference in chrominance and luminance between the current macroblock and the best match reference macroblock.

The operational functions of an MPEG-2 encoder are discussed in greater detail in commonly assigned, co-pending United States Patent Application Ser. No. 08/831,157, by Carr et al., filed Apr. 1, 1997, entitled "Control Scheme For Shared-Use Dual-Port Predicted Error Array," which is hereby incorporated herein by reference in its entirety.

As noted initially, encoder performance and/or picture quality may be enhanced in accordance with the principles of this invention through real-time video encoding. The video encoder is constructed to be adaptive to the video data received as a sequence of frames. Pursuant to this invention, a single encoding subsystem is employed in combination with pre-encode perceptual activity processing, which analyzes the video sequence prior to its real-time encoding. A delay buffer allows the perceptual activity measurement to be performed on frames of the video sequence prior to their encoding. The results are weighted and used to assign, for example, bit allocation for each frame in order to optimize picture quality. Analysis of the video sequence can comprise calculating one or more statistics from the video data.

The statistical measures can describe different characteristics of an image frame, for example, busyness of a frame, motion between image frames, scene change or fading, etc. Using the calculated statistics, adaptive encoding of the video sequence is carried out by controlling one or more encoding parameters of the real-time encoding process. For example, bit allocation, quantization parameter(s), encoding mode, etc., can be changed from frame to frame or macroblock to macroblock within a given frame according to derived statistics of a characteristic (e.g., scene content) of the particular frame(s).

One embodiment of an encoding system, generally denoted **500**, in accordance with the principles of this invention is depicted in FIG. **5**. The MPEG standard is again assumed herein for purposes of explanation; however, those skilled in the art will understand that other implementations and standards can employ the encoding concepts of this invention. System **500** includes a perceptual activity measurement unit **510**, variable bit rate logic **520**, a regulator **530** and a delay buffer **550**, which provides a delayed video sequence (pixel data) to an encoder **540**. Perceptual activity measurement unit **510**, variable bit rate logic **520** and regulator **530** cooperate to generate the desired statistics, such as inter-frame/intra-frame non-motion, motion, etc., statistics which are important to the encoder's **540** specific bit rate control algorithm. Encoder **540** generates, in one example, a variable bit rate encoded bitstream **541**.

Operationally, a sequence of video frames (i.e., digital video input) **501** is received and separated into control data **503** and pixel data **505**. Control data **503** is fed to regulator **530**, while pixel data **505** is forwarded to perceptual activity measurement unit **510** for analysis and to delay buffer **550**. The delay buffer can accommodate a predefined number of frames of digital data to be encoded. This delay provides the perceptual activity measurement unit **510** with time to analyze the data and produce statistics on the received signal. External microprocessor **511** selects the statistics to be produced by the perceptual activity measurement unit. External microprocessor **511** is used to represent the user interactivity feature of the describing system. It is a function

of VBR unit **520** to receive all required statistical information and inform encoder unit **540** (via regulator unit **530**) the encoded bit allocation, average MQUANT value, etc., to use in encoding a given picture's pixel data.

As described further below, the external processor's main function is to tell the perceptual activity measurement unit which calculations to perform and which results to send to variable bit rate (VBR) logic **520**. The perceptual activity measurement unit sends control signals to the VBR identifying which parameters are being sent. The variable bit rate logic assigns weighting factors to the PAM parameters to determine the number of bits to use for encoding the picture and any other relevant input parameters required by the encoder's rate control algorithm to officially carry out the picture encoding process.

Since the input video stream is buffered, statistical analysis can take longer than the available frame rate on a per-frame basis. For example, assume that the frame rate is $1/30^{th}$ of a second, and that frames 1, 2, 3 may be buffered. If frame 1 is analyzed in $1/300^{th}$ of a second, then $9/300^{th}$ of a second (i.e., 0.03 seconds) remains. Frame 2 can use this time for analysis while frame 1 is encoded, or a frame delay could be implemented so that frame 2 has $19/300^{th}$ of a second for analysis. This timing consideration can be readily implemented by one skilled in the art. By controlling timing, the external processor is provided with sufficient time to use processing power in the immediate past or immediate future frames, based upon current information, to enhance overall performance of the encode system.

Implementation of this invention can comprise software or hardware, or a combination thereof, configured to achieve the perceptual activity analysis goal disclosed herein. There are a wide variety of perceptual changes which can be detected, such as luminance, color, edges, motion, etc. Various perceptual analysis techniques are known in the art, any one of which can be employed within a system in accordance with this invention. Analysis can be accomplished by running video frames heavily laden with activity changes, and then those with little perceptual changes. The resultant data is analyzed for patterns which can then be used to select an encoding algorithm.

Returning to FIG. **5**, and to again summarize, digital video input **501** is received and analyzed for perceptual change activity, while the data is simultaneously buffered in delay buffer **550**. The perceptual activity statistics are passed to the VBR circuitry **520** where, for example, encoded picture bit allocation is performed. Thereafter, encoder **540** receives the delayed data and the statistics and performs the required encoding. Regulator **530** synchronizes the statistics generated by VBR unit **520** with the corresponding input picture fetched from delay buffer **550**, and monitors the encoder process to keep the variable bit rate stream **541** in sync with the frame speed.

Perceptual activity measurement unit **510**, variable bit rate logic **520** and regulator **530** in accordance with this invention are each described in greater detail below with references to FIGS. **6**–**8**.

FIG. **6** depicts an example of a perceptual activity measurement unit **510** which includes logic to evaluate luminance (shading) (**601**), chrominance (color) (**602**) and edges (**603**). The external microprocessor **511** can select the types of statistics to be calculated by the PAM unit.

By way of example, one detailed embodiment of edge detection logic **603** is described hereinbelow. Edges between arrangements of pixels within a macroblock can be detected as part of the spatial activity measurement, and then used in the calculation of perceived energy within a macroblock (i.e., calculation of perceptual MQUANT). Edge detection provides information and location of coefficients where a minimum quantization stepsize (i.e., MQUANT) is desirable. A large stepsize provides high compression, but creates more distortion. The opposite is true for a smaller stepsize. If there is a fine line in an image, a large stepsize may eliminate the line in the reconstructed image.

The desirability of employing edge detection when calculating perceptual MQUANT in an MPEG compliant video encoding system can be better understood from the following example. Selection of MQUANT in the conventional approach, described in the MPEG standard, is such that it uses a large MQUANT (large stepsize) for busy areas of video and small MQUANT (small stepsize) for non-active areas. This approach works well when the video sequence consists of only busy and non-busy blocks of pixels. This is because in the busy areas the distortion caused by using large stepsizes (large MQUANT) is masked by the image detail.

A significant drawback to the above-outlined approach is its deficiency in quantizing image blocks that contain hard edges. The conventional approach to adaptive quantization, classifies an edge-block as and therefore, a coarser quantization value is used. This results in severe artifacts in the edge areas and their neighboring pixels. Edges are image features that may extend over only a few pixels, yet are perceptually very important. An edge detection solution in accordance with this invention, therefore, identifies such image blocks and quantizes them for enhanced encoding. This is achieved by using a smaller MQUANT stepsize value for image blocks containing edges.

The identification of edge blocks by edge detection subunit **603** within the perceptual activity measurement unit can be performed by applying, for example, a Hadamard filter to pixel blocks within an image macroblock of a frame. One description of the Hadamard filter and its use in edge detection can be found in co-pending, commonly assigned United States Patent Application by Hall et al., entitled "Method and Apparatus For Adaptive Quantization," Ser. No. 08/618,659, filed Mar. 19, 1996, which is hereby incorporated herein by reference in its entirety. The main advantage of a Hadamard filter over other edge-detection methods is its ease of implementation. No multiplication is required in a Hadamard transform since the base Hadamard matrix consists of only 1's and −1's.

Based upon the above, edge detection processing as outlined herein improves upon the inherent masking approach imbedded in a conventional MPEG-2 perceptual MQUANT calculation. Again, edge detection is discussed by way of example only, and other perceptual activity changes can be readily monitored by those skilled in the art based upon the description presented herein.

Variable bit rate logic **520** receives picture content information (for example, edge detection information) from the perceptual activity measurement block, and group of picture (GOP) information, such as picture type, and number of B frames, as well as prior bit allocation information, from regulator **530** (FIG. **5**). Logic **520** uses this information to calculate numerous encoding parameters which the regulator block then sends to encoder **540** to facilitate maintenance of a constant quality, variable bit rate bitstream **541**.

The information (such as edge detection information) from the perceptual activity measurement unit is employed to predict the complexity of the current picture. The more complex the picture, the more bits are needed to insure that

the picture will be encoded at the same quality level as previous pictures. Thus, as noted, the regulator provides the variable bit rate logic with bit allocation information including group of picture structure, picture type, and number of bits used to encode the previous picture. These inputs to variable bit rate logic **520** are depicted in the flowchart of FIG. **7**, which presents one embodiment of certain processing implemented by perceptual activity measurement unit **510**, VBR logic **520** and regulator **530**.

As shown, perceptual activity measurement unit **510** receives a new picture input **700** and then tests for an edge **702**. Once calculated, the edge information is forwarded to the VBR unit **520**. Commensurate with receipt of this information, the VBR unit receives picture information **706** (as described above) from regulator **530**. The VBR logic uses the picture content information, for example, edge information **704**, and the picture type and encoding format information (i.e., picture information **706**), to determine a number of bits **708** the current picture should consume in order to maintain a substantially constant picture quality. This number of bits is provided to regulator **530** for use in control of the encoder **540** (FIG. **5**).

Allowable picture types are I, P and B pictures. Of these three picture types, the I picture type receives the most bits. This is due to the fact that P and B pictures will refer to I pictures for their motion estimation. Better quality on the I picture thus translates into better quality on P and B pictures. A P picture typically receives more bits than a B picture because it is used as a reference picture. The P picture refers to a previous I picture or to a previous P picture. The B picture typically has the fewest bits of the three picture types. The difference in bits used for picture type is principally achieved two ways, i.e., through motion estimation and/or through quantization.

Motion estimation is employed to find similarities between two pictures, with only the difference information being used. If the pictures are similar, then encoding the differences will use fewer bits than encoding the actual picture data. Skipping macroblocks is part of motion estimation, and allows parts of a picture that are exactly the same as their reference picture to go uncoded. The criteria for skipping macroblocks is more stringent for P pictures, again because of their use as reference pictures.

Quantization also controls bit usage. High quantization reduces the bits used, and also increases the amount of picture information lost in the encoding process. Quantization is scaled between three picture types, with I being the least quantized and B being the most quantized. Picture content information, along with picture type and GOP structure information, allows the VBR logic to accurately predict how many bits the current picture should consume to encode the picture with the same quality level as previous pictures in the video sequence. The quantization and motion estimation parameters comprise controllable parameters used in encoding the video sequence based upon the information derived by the pre-processing of the perceptual activity measurement unit and variable bit rate logic.

Regulator **530** (FIG. **5**) is the global data flow control and synchronization for the encode system **500**. The regulator manages two pipelines, i.e., input data and output data. This unit determines where to save each input picture data in delay buffer **550** and where to retrieve the data from for encoding by encoder **540**. Since the input picture order does not match the output picture order, the regulator is responsible for determining a memory location for each input picture to be buffered. The amount of memory required in the delay buffer is dictated by the size of the pictures and the quantity of pictures that are to be buffered.

As each picture data is received and analyzed, the determined encoding parameters are placed in a queue in the regulator, and subsequently retrieved when the associated picture is to be processed. The regulator has to determine when to start processing a picture, which input picture to process, where to retrieve the picture data, and where to retrieve its associated coding parameters to initiate a start to the encoder. One embodiment of this process is depicted in FIG. **8**. As shown, the regulator notes new input picture arrived **810** and determines a memory store location in delay buffer **550** for the frame **820**. Thereafter, the regulator initiates the perceptual activity measurement and variable bit rate processing described hereinabove **830**, and determines therefrom an encoded picture bit allocation **840** (i.e., an encoding parameter). The regulator simultaneously initiates storage of the received pixel data to the determined memory location in the delay buffer **850**, and thereafter controls sequencing **860** for the encoder. The regulator determines the memory fetch location and the coding parameter queue entry **870** and forwards the information to the encoder, along with a set picture type and set start_picture **880**.

Operationally, the regulator determines if the encoder system is ready to receive input picture data by monitoring for an appropriate output signal or other condition from the encoder. Using the IBM MPEG-2 encoder as an example, during power-on reset or when the encoder is paused, the encode system cannot receive input picture data and the INITCMP signal would be deactivated. When INITCMP is active, the regulator monitors the start of every input picture and determines the memory location in the delay buffer for storage. The regulator communicates the target memory location to the delay buffer and initiates the store process. The regulator also keeps a score of all unprocessed picture's status and location in memory. Each picture location is identified by a pointer. At the same time, the regulator initiates the start of perceptual activity measurements for the new picture. At the end of each input picture, the regulator expects coding parameters from the variable bit rate logic unit. The coding parameters are stored in a VBR queue within the regulator. The memory pointer is also saved in each queue entry and is used to associate the parameter with the picture data in memory.

The picture coding order is determined by availability of input picture, availability of coding parameters, GOP structure and availability of the encoder to accept these parameters. The GOP structure is selected by the user via the external microprocessor. With these inputs, the regulator determines when to start a picture through the coding process.

Those skilled in the art will note from the above discussion that encoding in accordance with the principles of the present invention will result in improved picture quality compared with conventional encoder systems. The encoding technique of this invention can insure a semi-constant picture quality of a decoded video sequence and constant bit rate (CBR) mode or a constant picture quality in variable bit rate (VBR) encoding mode.

The present invention can be included, for example, in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The articles manufactured can be included as part of the computer system or sold separately.

The flow diagrams depicted herein are provided by way of example. There may be variations to these diagrams or the steps or operations described herein without departing from the spirit of the invention. for instance, in certain cases the steps may be performed in differing order, or steps may be added, deleted or modified. All these variations are considered to comprise part of the present invention as recited in the appended claims.

While the invention has been described in detail herein in accordance with certain preferred embodiments thereof, many modifications and changes therein may be effected by those skilled in the art. Accordingly, it is intended by the appended claims to cover all such modifications and changes as fall within the true spirit and scope of the invention.

What is claimed is:

1. A method for encoding a sequence of video frames using a single encoding engine, said method comprising:

(a) performing perceptual activity analysis on a frame of the sequence of video frames to derive information on at least one characteristic thereof;

(b) setting at least one controllable parameter for use in encoding said frame of the sequence of video frames based upon said information derived on said at least one characteristic thereof; and

(c) encoding said frame of the sequence of video frames in real-time using said single encoding engine and said at least one controllable parameter.

2. The method of claim **1**, wherein said at least one controllable parameter of said setting (b) comprises at least one of a quantization parameter or a motion estimation parameter for use by said single encoding engine in encoding said frame of said sequence of video frames.

3. The method of claim **1**, wherein said performing (a) comprises performing perceptual activity analysis on said frame of the sequence of video frames to derive information on at least one of shading, scene change, fade, color, motion or edge presence within said frame.

4. The method of claim **1**, wherein said performing (a) comprises performing perceptual activity analysis on at least one subdivision, picture region, slice, macroblock or block within said frame of said sequence of video frames to derive information on said at least one characteristic thereof.

5. The method of claim **1**, further comprising repeating said performing (a), said setting (b) and said encoding (c) in pipeline fashion for a plurality of frames of said sequence of video frames.

6. The method of claim **5**, further comprising for each frame of said plurality of frames buffering said frame during said performing (a) and said setting (b), and wherein said method further comprises regulating said buffering and initiating of said encoding (c) to insure synchronization of encoding of each frame of said plurality of frames with the at least one controllable parameter set therefore.

7. The method of claim **6**, wherein for each frame of said plurality of frames, said regulating includes controlling storing of said frame in a delay buffer during said performing (a) and said setting (b) and retrieving of said frame from the delay buffer prior to said encoding (c).

8. The method of claim **5**, wherein said encoding (c) produces a variable bit rate bitstream from said plurality of frames of the sequence of video frames, and wherein said method comprises performing said steps (a)–(c) to maintain substantially constant quality in said variable bit rate bitstream.

9. The method of claim **8**, wherein for each frame of said plurality of frames, said setting (b) comprises determining a number of bits to use in said encoding (c) of said frame, said determining of said number of bits to use being based on at least some of group of picture (GOP) information, picture type, number of B pictures between reference pictures in said sequence of video frames, bit allocation of at least one prior encoded frame of the sequence of video frames, and said derived information on said at least one characteristic of said frame.

10. The method of claim **1**, wherein said performing (a) comprises performing perceptual activity analysis on a frame of the sequence of video frames to derive information on multiple characteristics thereof, and wherein said method further comprises for said frame, selecting among said information on said multiple characteristics thereof and providing said selected information to said setting (b).

11. The method of claim **10**, wherein said selecting comprises weighting said information on each of said multiple characteristics, and said setting (b) comprises setting said at least one controllable parameter based upon said weighted information.

12. The method of claim **10**, wherein said performing (a) comprises performing perceptual activity analysis on said frame of the sequence of video frames to derive information on at least some of shading, scene change, fade, color, motion and edge presence within said frame.

13. The method of claim **1**, wherein said performing (a) comprises performing perceptual activity analysis on said frame of the sequence of video frames to derive information on edge presence within said frame, and upon said performing (a) detecting an edge within said frame, said setting (b) comprising setting a quantization parameter for a small stepsize, said quantization parameter comprising said at least one controllable parameter and being employed by said encoding (c) during encoding of said frame of the sequence of video frames.

14. The method of claim **1**, wherein said setting (b) comprises setting multiple controllable parameters based upon said information derived on said at least one characteristic of said frame, and wherein said encoding (c) comprises encoding said frame of the sequence of video frames in real-time using said single encoding engine and said multiple controllable parameters.

15. A system for encoding a sequence of video frames comprising:

a pre-encode processing unit, said pre-encode processing unit comprising:

a perceptual activity measurement unit for performing perceptual activity analysis on a frame of the sequence of video frames to derive information on at least one characteristic thereof;

a variable bit rate unit for setting at least one controllable parameter based upon said information derived on said at least one characteristic thereof, said at least one controllable parameter to be used in encoding said frame of the sequence of video frames; and

a single encoding engine for encoding said frame of the sequence of video frames in real-time using the at least one controllable parameter set via said pre-encode processing unit.

16. The system of claim **15**, wherein said at least one controllable parameter comprises at least one of a quantization parameter or a motion estimation parameter for use by said single encoding engine in encoding said frame of said sequence of video frames.

17. The system of claim **15**, wherein said perceptual activity measurement unit comprises logic for performing perceptual activity analysis on said frame of the sequence of video frames to derive information on at least one of

shading, scene change, fade, color, motion or edge presence within said frame.

18. The system of claim **15**, wherein said pre-encode processing unit and said single encoding engine comprise means for pipeline processing a plurality of frames of said sequence of video frames.

19. The system of claim **18**, further comprising a delay buffer and a regulator, said regulator coordinating for each frame of said plurality of frames: buffering of said frame in said delay buffer during processing of said perceptual activity measurement unit and said variable bit rate unit; and initiating said single encoding engine to encode said frame in synchronization with said at least one controllable parameter set by said variable bit rate unit.

20. The system of claim **15**, wherein said single encoding engine comprises an MPEG compliant encoder, said MPEG compliant encoder producing a variable bit rate bitstream, and wherein said pre-encode processing unit comprises means for maintaining substantially constant quality in said variable bit rate bitstream output by said MPEG compliant encoder.

21. The system of claim **15**, wherein said at least one controllable parameter comprises a number of bits to use by said single encoding engine in encoding said frame of the sequence of frames, and wherein said variable bit rate unit comprises means for determining said number of bits to use based on at least some of group of picture (GOP) information, picture type, number of B pictures between reference pictures in said sequence of video frames, bit allocation for at least one prior encoded frame of the sequence of video frames, and said derived information on said at least one characteristic of said frame.

22. The system of claim **15**, wherein said perceptual activity measurement unit comprises means for deriving information on multiple characteristics of said frame of said sequence of video frames, and wherein said system further comprises processing means for weighting information on each of said multiple characteristics, said variable bit rate unit setting said at least one controllable parameter based on said weighted information.

23. The system of claim **15**, wherein said perceptual activity measurement unit comprises means for performing perceptual activity analysis on said frame of the sequence of video frames to derive information on edge presence within said frame, and upon detection of an edge within the frame, said variable bit rate unit comprises means for setting a quantization parameter for a small stepsize, said quantization parameter comprising said at least one controllable parameter employed by said single encoding engine during encoding of said frame of the sequence of video frames.

24. A computer program product comprising a computer usable medium having computer readable program code means therein for use in encoding a sequence of video frames using a single encoding engine, said computer readable program code means in said computer program product comprising:

- computer readable program code means for causing a computer to affect performing perceptual activity analysis on a frame of the sequence of video frames to derive information on at least one characteristic thereof;
- computer readable program code means for causing a computer to affect setting at least one controllable parameter for use in encoding said frame of the sequence of video frames based upon said information derived on said at least one characteristic thereof; and
- computer readable program code means for causing a computer to affect encoding of said frame of said sequence of video frames in realtime using said single encoding engine and the at least one controllable parameter.

25. The computer readable program code means of claim **24**, wherein said at least one controllable parameter comprises at least one of a quantization parameter or a motion estimation parameter for use by said single encoding engine in encoding said frame of said sequence of video frames.

26. The computer readable program code means of claim **24**, wherein said computer readable program code means for causing a computer to affect performing perceptual activity analysis on said frame comprises computer readable program code means for causing a computer to affect performing perceptual activity analysis on said frame of the sequence of video frames to derive information on at least one of shading, scene change, fade, color, motion or edge presence within said frame.

27. The computer readable program code means of claim **24**, further comprising computer readable program code means for causing a computer to affect regulating of said performing of perceptual activity analysis on the frame and said setting of at least one controllable parameter, said computer readable program code means for causing a computer to affect regulating comprising computer readable program code means for causing a computer to affect buffering of said frame in a delay buffer and subsequent initiating of said encoding to insure synchronization of encoding of said frame with the at least one controllable parameter set therefore.

28. The computer readable program code means of claim **24**, wherein said computer readable program code means for causing a computer to affect encoding of said frame produces a variable bit rate bitstream, and wherein said computer readable program code means further comprises computer readable program code means for maintaining substantially constant quality in said variable bit rate bitstream.

29. The computer readable program code means of claim **28**, wherein said computer readable program code means for causing a computer to affect said setting of at least one control parameter comprises for each frame, computer readable program code means for causing a computer to affect determining of a number of bits to use in encoding said frame, said determining of said number of bits to use being based on at least some of group of picture (GOP) information, picture type, number of B pictures between reference pictures in said sequence of video frames, bit allocation for at least one prior encoded frame of the sequence of video frames, and said derived information on said at least one characteristic of said frame.

30. The computer readable program code means of claim **24**, wherein said computer readable program code means for causing a computer to affect performing perceptual activity analysis comprises computer readable program code means for causing a computer to affect performing perceptual activity analysis on said frame of the sequence of video frames to derive information on multiple characteristics thereof, and wherein said computer readable program code means further comprises computer readable program code means for causing a computer to affect selecting among said information on said multiple characteristics of said frame and providing said selected information to said computer readable program code means for causing a computer to affect setting of said at least one controllable parameter.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,097,757

DATED : August 1, 2000

INVENTOR(S) : Boice et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

IN THE TITLE:

Delete "EMPLOYING STATISTICS" and replace with --EMPLOYING IMAGE STATISTICS--.

IN THE SPECIFICATION:

Col. 2, line 41, delete "This because" and replace with --This is because--.

Col. 2, line 63, delete "ith" and replace with --i^{th}--.

Col. 3, line 1, delete "ith" and replace with --i^{th}--.

Col. 4, line 55, delete "ith" and replace with --i^{th}--.

Col. 5, line 5, delete "generated" and replace with --once generated--.

Col. 7, line 9, delete "VBR identi" and replace with --VBR logic identi--.

Col. 8, line 24, delete "as and" and replace with --as busy, and--.

Signed and Sealed this

Twenty-fourth Day of April, 2001

Attest:

Nicholas P. Godici

NICHOLAS P. GODICI

Attesting Officer

Acting Director of the United States Patent and Trademark Office

VIEW BASKET

AskOxford.com
Oxford Dictionaries. Passionate about language

SEARCH
Go
Entire AskOxford Site

HOME · SHOP · EDUCATION · PRESS ROOM · CONTACT US ·
ASK THE EXPERTS · BETTER WRITING · WORLD OF WORDS · GAMES · GLOBAL ENGLISH · FOREIGN LANGUAGES

SELECT VIEW UK US
You are currently in the UK view

Compact Oxford English Dictionary

firmware

• **noun** Computing permanent software programmed into a read-only memory.

Perform another search of the Compact Oxford English Dictionary

About this dictionary
The *Compact Oxford English Dictionary of Current English* contains 145,000 words, phrases, and definitions.
Find out more about Oxford's range of English dictionaries
Sign up for the AskOxford Word of the Day

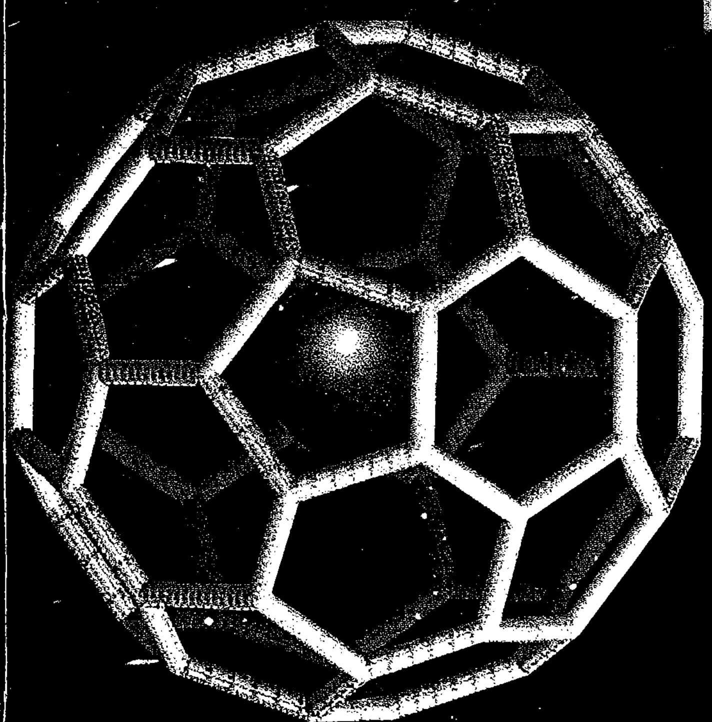
Search the Little Oxford Dictionary of Quotations
Search the Concise Dictionary of First Names

- Ask The Experts
- Better Writing
- World of Words
- Games
- Global English
- Foreign Languages

LINKS
- AskOxford Shop
- External Web Links
- OUP Website
- Children's Dictionaries
- ELT Dictionaries
- Oxford Online

OXFORD UNIVERSITY PRESS

McGraw-Hill Dictionary of SCIENTIFIC and TECHNICAL TERMS



Sixth Edition

On the cover: Representation of a fullerene molecule with a noble gas atom trapped inside. At the Permian-Triassic sedimentary boundary the noble gases helium and argon have been found trapped inside fullerenes. They exhibit isotope ratios quite similar to those found in meteorites, suggesting that a fireball meteorite or asteroid exploded when it hit the Earth, causing major changes in the environment. *(Image copyright © Dr. Luann Becker. Reproduced with permission.)*

Over the six editions of the Dictionary, material has been drawn from the following references: G. M. Garrity et al., *Taxonomic Outline of the Procaryotes,* Release 2, Springer-Verlag, January 2002; D. W. Linzey, *Vertebrate Biology,* McGraw-Hill, 2001; J. A. Pechenik, *Biology of the Invertebrates,* 4th ed., McGraw-Hill, 2000; *U.S. Air Force Glossary of Standardized Terms,* AF Manual 11-1, vol. 1, 1972; F. Casey, ed., *Compilation of Terms in Information Sciences Technology,* Federal Council for Science and Technology, 1970; *Communications-Electronics Terminology,* AF Manual 11-1, vol. 3, 1970; P. W. Thrush, comp. and ed., *A Dictionary of Mining, Mineral, and Related Terms,* Bureau of Mines, 1968; *A DOD Glossary of Mapping, Charting and Geodetic Terms,* Department of Defense, 1967; J. M. Gilliland, *Solar-Terrestrial Physics: A Glossary of Terms and Abbreviations,* Royal Aircraft Establishment Technical Report 67158, 1967; W. H. Allen, ed., *Dictionary of Technical Terms for Aerospace Use,* National Aeronautics and Space Administration, 1965; *Glossary of Stinfo Terminology,* Office of Aerospace Research, U.S. Air Force, 1963; *Naval Dictionary of Electronic, Technical, and Imperative Terms,* Bureau of Naval Personnel, 1962; R. E. Huschke, *Glossary of Meteorology,* American Meteorological Society, 1959; *ADP Glossary,* Department of the Navy, NAVSO P-3097; *Glossary of Air Traffic Control Terms,* Federal Aviation Agency; *A Glossary of Range Terminology, White Sands Missile Range, New Mexico,* National Bureau of Standards, AD 467-424; *Nuclear Terms: A Glossary,* 2d ed., Atomic Energy Commission.

McGRAW-HILL DICTIONARY OF SCIENTIFIC AND TECHNICAL TERMS, Sixth Edition

2 3 4 5 6 7 8 9 0 DOW/DOW 0 8 7 6 5 4 3

ISBN 0-07-042313-X

Library of Congress Cataloging-in-Publication Data

McGraw-Hill dictionary of scientific and technical terms--6th ed.
p. cm.
ISBN 0-07-042313-X (alk. paper)
1. Science--Dictionaries. 2. Technology--Dictionaries. I. Title: Dictionary of scientific and technical terms.

firing mechanism [ENG] A mechanism for firing a primer; the primer may be for initiating the propelling charge, in which case the firing mechanism forms a part of the weapon; if the primer is for the purpose of initiating detonation of the main charge, the firing mechanism is a part of the ammunition item and performs the function of a fuse. { 'fīr·iŋ ,mek·ə,niz·əm }

firing pin [ORD] A device used in the firing mechanism of a gun, mine, bomb, fuse, projectile, or the like, which strikes and detonates a sensitive explosive to initiate an explosive train or a propelling charge. { 'fīr·iŋ ,pin }

firing point [ELECTR] *See* critical grid voltage. [ORD] Location from which fire is delivered in target practice. { 'fīr·iŋ ,pȯint }

firing position [ORD] Position of a weapon ready for firing. { 'fīr·iŋ pə,zish·ən }

firing potential [ELECTR] Controlled potential at which conduction through a gas-filled tube begins. { 'fīr·iŋ pə,ten·chəl }

firing pressure [MECH ENG] The highest pressure in an engine cylinder during combustion. { 'fīr·iŋ ,presh·ər }

firing rate [MECH ENG] The rate at which fuel feed to a burner occurs, in terms of volume, heat units, or weight per unit time. { 'fīr·iŋ ,rāt }

firing table [ORD] A table or chart giving data needed for firing a gun accurately on a target under standard conditions, and also the corrections that must be made for special conditions such as winds or variations of temperature. { 'fīr·iŋ ,tā·bəl }

firing table elevation [ORD] The angle between the axis of the bore and the horizontal plane when the piece is laid to fire at a given range under conditions that are accepted as standard. { 'fīr·iŋ ,tā·bəl el·ə'vā·shən }

firing time [ORD] The period of time during which a weapon is fired. { 'fīr·iŋ ,tīm }

firmer chisel [DES ENG] A small hand chisel with a flat blade; used in woodworking. { 'fər·mər ,chiz·əl }

firming agent [FOOD ENG] A substance added prior to or during the processing of a foodstuff to protect and retain natural firmness. { 'firm·iŋ ,ā·jənt }

firm-joint caliper [DES ENG] An outside or inside caliper whose legs are jointed together at the top with a nut and which must be opened and closed by hand pressure. { 'fərm ,jȯint 'kal·ə·pər }

firmoviscosity [MECH] Property of a substance in which the stress is equal to the sum of a term proportional to the substance's deformation, and a term proportional to its rate of deformation. { ¦fər·mō·vis¦käs·əd·ē }

firmware [COMPUT SCI] A computer program or instruction, such as a microprogram, used so often that it is stored in a read-only memory instead of being included in software; often used in computers that monitor production processes. { 'fərm,wer }

firn [HYD] Material transitional between snow and glacier ice; it is formed from snow after existing through one summer melt season and becomes glacier ice when its permeability to liquid water drops to zero. Also known as firn snow. { fərn }

firn basin *See* firn field. { 'fərn ,bās·ən }

fir needle oil [MATER] An essential oil distilled from the needles and twigs of some trees of the genus *Abies;* used in perfumery, flavoring, and medicine. Also known as fir oil. { 'fər ,nēd·əl ,ȯil }

firn field [HYD] The accumulation area or upper region of a glacier where snow accumulates and firn is secreted. Also known as firn basin. { 'fərn ,fēld }

firn ice *See* iced firn. { 'fərn ,īs }

firnification [HYD] The process of firn formation from snow and of transformation of firn into glacier ice. { ,fər·nə·fə'kā·shən }

firn limit *See* firn line. { 'fərn ,lim·ət }

firn line [GEOL] **1.** The regional snow line on a glacier. **2.** The line that divides the ablation area of a glacier from the accumulation area. Also known as firn limit. { 'fərn ,līn }

firn snow *See* firn; old snow. { 'fərn ,snō }

fir oil *See* fir needle oil. { 'fər ,ȯil }

first answer print [GRAPHICS] The motion picture composite print, with the mixed sound track and the optical effects. { ¦fərst 'an·sər ,print }

first arrival [ENG] In exploration refraction seismology, the first seismic event recorded on a seismogram; it is noteworthy in that only first arrivals are considered in this usage. { ¦fərst ə'rī·vəl }

first bottom [GEOL] The floodplain of a river, below the first terrace. { ¦fərst 'bäd·əm }

first category [MATH] **1.** A set is of first category if it is a countable union of nowhere dense sets. **2.** A set S is of first category at a point x if there is a neighborhood of x whose intersection with S is of first category. { ¦fərst 'kad·ə,gȯr·ē }

first-class current [PART PHYS] A weak-interaction current whose charge symmetry (or G parity) properties are the same as those of currents which arise in the Fermi theory of beta decay. { ¦fərst ,klas 'kə·rənt }

first cost [IND ENG] The sum of the initial expenditures involved in capitalizing a property; includes items such as transportation, installation, preparation for service, as well as other related costs. { ¦fərst 'kȯst }

first countable topological space [MATH] A topological space in which every point has a countable number of open neighborhoods so that any neighborhood of this point contains one of these. { ¦fərst 'kaunt·ə·bəl ,täp·ə¦läj·ə·kəl 'spās }

first-degree burn [MED] A mild burn characterized by pain and reddening of the skin. { ¦fərst də¦grē 'bərn }

first derivative [MATH] The derivative of a function, considered as a function of the independent variable just as was the original function from which the derivative was taken. { ¦fərst də'riv·əd·iv }

first derived curve *See* derived curve. { ¦fərst də¦rīvd 'kərv }

first detector *See* mixer. { ¦fərst di'tek·tər }

first estimate-second estimate method [NAV] The method of determining time of meridian transit (especially local apparent noon) for a moving craft; the time of transit is computed for an estimated longitude of the craft, and the longitude estimate is then revised to agree with the time determined by the first estimate, and a second computation is made; the process is repeated as many times as necessary to obtain an answer of the desired precision. { ¦fərst ,es·tə·mət 'sek·ənd ,es·tə·mət ,meth·əd }

first filial generation [GEN] The first generation resulting from a cross between parents homozygous for different alleles at a locus; all members are heterozygous at the locus. Symbolized F_1. { ¦fərst ¦fil·ē·əl jen·ə'rā·shən }

first fire [ENG] The igniter used with pyrotechnic devices, consisting of first fire composition, loaded in direct contact with the main pyrotechnic charge; the ignition of the igniter or first fire is generally accomplished by fuse action. { ¦fərst 'fīr }

first fire composition [MATER] A pyrotechnic composition (readily ignitable and easily pressed into a strong, solid mass), compounded to produce a high temperature, preferably with creation of slag to give heat capacity. { ¦fərst ¦fīr ,käm·pə'zish·ən }

first Fresnel zone [ELECTROMAG] Circular portion of a wavefront transverse to the line between an emitter and a more distant point, where the resultant disturbance is being observed, whose center is the intersection of the front with the direct ray, and whose radius is such that the shortest path from the emitter through the periphery to the receiving point is one-half wavelength longer than the direct ray. { ¦fərst frə'nel ,zōn }

first-generation [COMPUT SCI] Denoting electronic hardware, logical organization, and software characteristic of a first-generation computer. { ¦fərst jen·ə'rā·shən }

first-generation computer [COMPUT SCI] A computer from the earliest stage of computer development, ending in the early 1960s, characterized by the use of vacuum tubes, the performance of one operation at a time in strictly sequential fashion, and elementary software, usually including a program loader, simple utility routines, and an assembler to assist in program writing. { ¦fərst jen·ə¦rā·shən kəm'pyüd·ər }

first gust [METEOROL] The sharp increase in wind speed often associated with the early mature stage of a thunderstorm cell; it occurs with the passage of the discontinuity zone which is the boundary of the cold-air downdraft. { ¦fərst 'gəst }

first harmonic *See* fundamental. { ¦fərst här'män·ik }

first-in, first-out [IND ENG] An inventory cost evaluation method which transfers costs of material to the product in chronological order. Abbreviated FIFO. { ¦fərst 'in ¦fərst 'aut }

first-item list [COMPUT SCI] A series of records that is

illuvial [GEOL] Pertaining to a region or material characterized by the accumulation of soil by the illuviation of another zone or material. { iˈlü·vē·əl }

illuvial horizon *See* B horizon. { iˈlü·vē·əl həˈrīz·ən }

illuviation [GEOL] The deposition of colloids, soluble salts, and small mineral particles in an underlying layer of soil. { i,lü·vēˈā·shən }

illuvium [GEOL] Material leached by chemical or other processes from one soil horizon and deposited in another. { iˈlü·vē·əm }

ilmenite [MINERAL] $FeTiO_3$ An iron-black, opaque, rhombohedral mineral that is the principal ore of titanium. Also known as mohsite; titanic iron ore. { il·mə,nīt }

ILS *See* instrument landing system.

ilsemannite [MINERAL] A black, blue-black, or blue mineral composed of hydrous molybdenum oxide or perhaps sulfate, occurring in earthy massive form. { ˈil·sə·mə,nīt }

ILS reference point [NAV] A point that is on the centerline of a runway served by an ILS and that is designated as the optimum point of contact for the landing aircraft. Derived from instrument landing system reference point. { ˈīˈelˈes ˈref·rəns ,pȯint }

image [ACOUS] *See* acoustic image. [COMMUN] **1.** One of two groups of side bands generated in the process of modulation; the unused group is referred to as the unwanted image. **2.** The scene reproduced by a television or facsimile receiver. [COMPUT SCI] A copy of the information contained in one medium recorded on a different data medium. [ELEC] *See* electric image. [ELECTROMAG] The input reflection coefficient corresponding to the reflection coefficient of a specified load when the load is placed on one side of a waveguide junction and a slotted line is placed on the other. [MATH] **1.** For a point x in the domain of a function f, the point $f(x)$. **2.** For a subset A of the domain of a function f, the set of all points that are equal to $f(x)$ for some point x in A. [OPTICS] An optical counterpart of a self-luminous or illuminated object formed by the light rays that traverse an optical system; each point of the object has a corresponding point in the image from which rays diverge or appear to diverge. [PHYS] Any reproduction of an object produced by means of focusing light, sound, electron radiation, or other emanations coming from the object or reflected by the object. [PSYCH] A representation of a sensory experience, occurring in the brain. { ˈim·ij }

image admittance [ELECTR] The reciprocal of image impedance. { ˈim·ij ad,mit·əns }

image antenna [ELECTROMAG] A fictitious electrical counterpart of an actual antenna, acting mathematically as if it existed in the ground directly under the real antenna and served as the direct source of the wave that is reflected from the ground by the actual antenna. { ˈim·ij anˈten·ə }

image attenuation constant [ELECTR] The real part of the image transfer constant. { ˈim·ij ə,ten·yəˈwā·shən ,kän·stənt }

image converter [ELECTR] *See* image tube. [OPTICS] A converter that uses a fiber optic bundle to change the form of an image, for more convenient recording and display or for the coding of secret messages. { ˈim·ij kən,vərd·ər }

image converter camera [ELECTR] A camera consisting of an image tube and an optical system which focuses the image produced on the phosphorescent screen of the tube onto photographic film. { ˈim·ij ,kən,vərd·ər ,kam·rə }

image dissection photography [ELECTR] A method of high-speed photography in which an image is split in any one of various ways into interlaced space and time elements which can be unscrambled or played back through the system either to be viewed or to give a master negative. { ˈim·ij di,sek·shən fə,täg·rə·fē }

image dissector [COMPUT SCI] In optical character recognition, a device that optically examines an input character for the purpose of breaking it down into its prescribed elements. { ˈim·ij di,sek·tər }

image dissector tube [ELECTR] A television camera tube in which an electron image produced by a photoemitting surface is focused in the plane of the defining aperture and is scanned past that aperture. Also known as Farnsworth image dissector tube. { ˈim·ij di,sek·tər ,tüb }

image effect [ELECTROMAG] Effect produced on the field of an antenna due to the presence of the earth; electromagnetic waves are reflected from the earth's surface, and these reflections often are accounted for by an image antenna at an equal distance below the earth's surface. { ˈim·ij i,fekt }

image enhancement [COMPUT SCI] Improvement of the quality of a picture, with the aid of a computer, by giving it higher contrast or making it less blurred or less noisy. { ˈim·ij inˈhans·mənt }

image force [ELEC] The electrostatic force on a charge in the neighborhood of a conductor, which may be thought of as the attraction to the charge's electric image. { ˈim·ij ,fȯrs }

image frequency [ELECTR] An undesired carrier frequency that differs from the frequency to which a superheterodyne receiver is tuned by twice the intermediate frequency. { ˈim·ij ,frē·kwən·sē }

image iconoscope [ELECTR] A camera tube in which an optical image is projected on a semitransparent photocathode, and the resulting electron image emitted from the other side of the photocathode is focused on a separate storage target; the target is scanned on the same side by a high-velocity electron beam, neutralizing the elemental charges in sequence to produce the camera output signal at the target. Also known as superemitron camera (British usage). { ˈim·ij īˈkän·ə,skōp }

image impedance [ELECTR] One of the impedances that, when connected to the input and output of a transducer, will make the impedances in both directions equal at the input terminals and at the output terminals. { ˈim·ij im,pēd·əns }

image intensifier *See* light amplifier. { ˈim·ij inˈten·sə,fī·ər }

image interference [COMMUN] Interference occurring in a superheterodyne receiver when a station broadcasting on the image frequency is received along with the desired station. { ˈim·ij ,in·tərˈfir·əns }

image isocon [ELECTR] A television camera tube which is similar to the image orthicon but whose return beam consists of scanning beam electrons that are scattered by positive stored charges on the target. { ˈim·ij ˈī·sə,kän }

image load [ELECTR] Load parameters reflected back to the source by line discontinuities. { ˈim·ij ,lōd }

image orthicon [ELECTR] A television camera tube in which an electron image is produced by a photoemitting surface and focused on one side of a separate storage tube that is scanned on its opposite side by a beam of low-velocity electrons; electrons that are reflected from the storage tube, after positive stored charges are neutralized by the scanning beam, form a return beam which is amplified by an electron multiplier. { ˈim·ij ˈȯr·thə,kän }

image parameter design [ELECTR] A method of filter design using image impedance and image transfer functions as the fundamental network functions. { ˈim·ij pəˈram·əd·ər di,zīn }

image parameter filter [ELECTR] A filter constructed by image parameter design. { ˈim·ij pəˈram·əd·ər ,fil·tər }

image phase constant [ELECTR] The imaginary part of the image transfer constant. { ˈim·ij ˈfāz ,kän·stənt }

image plane [OPTICS] The plane in which an image produced by an optical system is formed; if the object plane is perpendicular to the optical axis, the image plane will ordinarily also be perpendicular to the axis. { ˈim·ij ,plān }

image potential [ELEC] The potential set up by an electric image. { ˈim·ij pə,ten·chəl }

image processing [COMPUT SCI] A technique in which the data from an image are digitized and various mathematical operations are applied to the data, generally with a digital computer, in order to create an enhanced image that is more useful or pleasing to a human observer, or to perform some of the interpretation and recognition tasks usually performed by humans. Also known as picture processing. { ˈim·ij ,prä·ses·iŋ }

image ratio [ELECTR] In a heterodyne receiver, the ratio of the image frequency signal input at the antenna to the desired signal input for identical outputs. { ˈim·ij ,rā·shō }

image reject mixer [ELECTR] Combination of two balanced mixers and associated hybrid circuits designed to separate the image channel from the signal channels normally present in a conventional mixer; the arrangement gives image rejection up to 30 decibels without the use of filters. { ˈim·ij ˈrē,jekt ,mik·sər }

image response [ELECTR] The response of a superheterodyne receiver to an undesired signal at its image frequency. { ˈim·ij ri,späns }

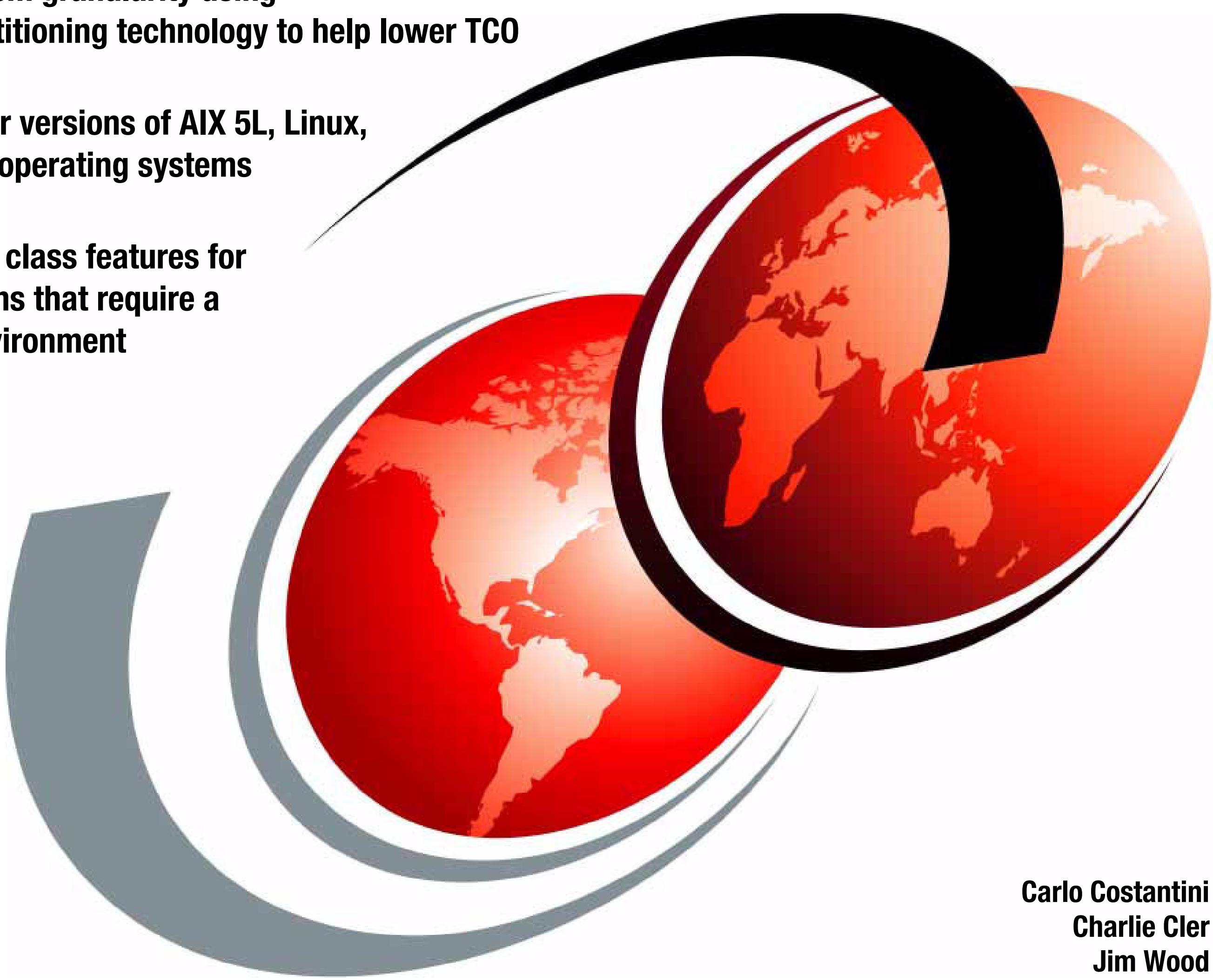
IBM

IBM System p5 590 and 595
Technical Overview and Introduction

Finer system granularity using Micro-Partitioning technology to help lower TCO

Support for versions of AIX 5L, Linux, and i5/OS operating systems

Enterprise class features for applications that require a robust environment



Carlo Costantini
Charlie Cler
Jim Wood

Redpaper

ibm.com/redbooks

International Technical Support Organization

IBM System p5 590 and 595 Technical Overview and Introduction

September 2006

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

Second Edition (September 2006)

This edition applies to the IBM System p5 590 and 595, and AIX 5L Version 5.3, product number 5765-G03.

```
frequency 210000000     Processor Speed       False
smt_enabled true        Processor SMT enabled False
smt_threads 2           Processor SMT threads False
state enable            Processor state       False
type powerPC_POWER5     Processor type        False
```

(False, as used in this output, signifies that the value cannot be changed through an AIX 5L command interface.)

pmcycles -m This command (AIX 5L Version 5.3 and later) uses the performance monitor cycle counter and the processor real-time clock to measure the actual processor clock speed in MHz. This is the sample output of a 16-core p5-590 running at 2.1 GHz:

```
Cpu 0 runs at 2100 MHz
Cpu 1 runs at 2100 MHz
Cpu 2 runs at 2100 MHz
Cpu 3 runs at 2100 MHz
...
Cpu 13 runs at 2100 MHz
Cpu 14 runs at 2100 MHz
Cpu 15 runs at 2100 MHz
```

2.2 System flash memory configuration

In the p5-590 and p5-595, a serial electronically erasable programmable read-only memory (sEEPROM) adapter plugs into the back of the Central Electronics Complex backplane. The platform firmware binary image is programmed into the system's EEPROM, also known as *system FLASH memory*. FLASH memory is initially programmed during manufacturing of the p5-590 and p5-595 systems. However, this single binary image can be reprogrammed to accommodate firmware fixes provided to the client using the hardware management console.

The firmware binary image contains boot code for the p5-590 and p5-595. This boot code includes, but is not limited to, system service processor code; code to initialize the POWER5+ or POWER5 processors, memory, and I/O subsystem components; partition management code; and code to support the virtualization features. The firmware binary image also contains hardware monitoring code used during partition run time.

During boot time, the system service processor dynamically allocates the firmware image from flash memory into main system memory. The firmware code is also responsible for loading the operating system image into main memory.

2.2.1 Vital product data and system smart chips

Vital product data (VPD) carries all of the necessary information for the service processor to determine if the hardware is compatible and how to configure the hardware and system processors on the card. The VPD also contains the part number and serial number of the card used for servicing the machine, as well as the location information of each device for failure analysis. Because the VPD in the card carries all information necessary to configure the card, no card device drivers or special code have to be sent with each card for installation.

Smart chips are micro-controllers used to store vital product data (VPD). The smart chip provides a means for securely storing data that cannot be read, altered, or written other than by IBM privileged code. The smart chip provides a means of verifying IBM System On/Off Capacity on Demand and IBM System Capacity Upgrade on Demand activation codes that

[1] The output of the **lsattr** command has been expanded with AIX 5L to include the processor clock rate.

Redbooks Paper

Jon Tate
Cameron Hildebran

L10 Implementation

Overview

This paper details the basic installation, configuration, and use of an entry SAN switch with the IBM® TotalStorage® 2006 model L10 switch. In less complex SAN environments, with fewer servers and storage arrays, single switch or dual cascaded switches offer redundancy and performance with minimal administration and lower cost than larger directors. For this example we use two IBM Thinkcentre machines running Windows® as host servers. To connect to the L10 switches utilized by the Windows servers use QLogic QLA2300 HBAs. Storage is supplied by an IBM TotalStorage DS400 with six 36.4 GB disk drives.

Before you begin

Make sure that you:

- Have IP addresses ready to assign to the L10 switches and the DS400 controllers.
- Have Ethernet cables ready to integrate the SAN with your network switch. This is the switch the servers are connected to and will be used in the SAN installation.
- Know the servers' host names (defined during server installation) and be ready to assign a name to the SAN.
- Have an unzip utility such as PKZIP or WinZip.

If installing the DS400 and L10 switches in a rack, follow the guide *Rack Installation Instructions* for the DS400 system, and *2006 model L10 Fixed Rack-Mount Kit Instructions* for the L10 switches. These instructions accompany the DS400 system and the L10 switch.

Note: Do not turn on power to the hardware at this point.

Install HBAs

This step assumes that the servers are already configured with the operating system and are connected to the network switch according to the appropriate network settings for your environment.

f. Click **Add**. The storage will now be visible in the right side of the ServeRAID Manager program interface, as shown in Figure 22.

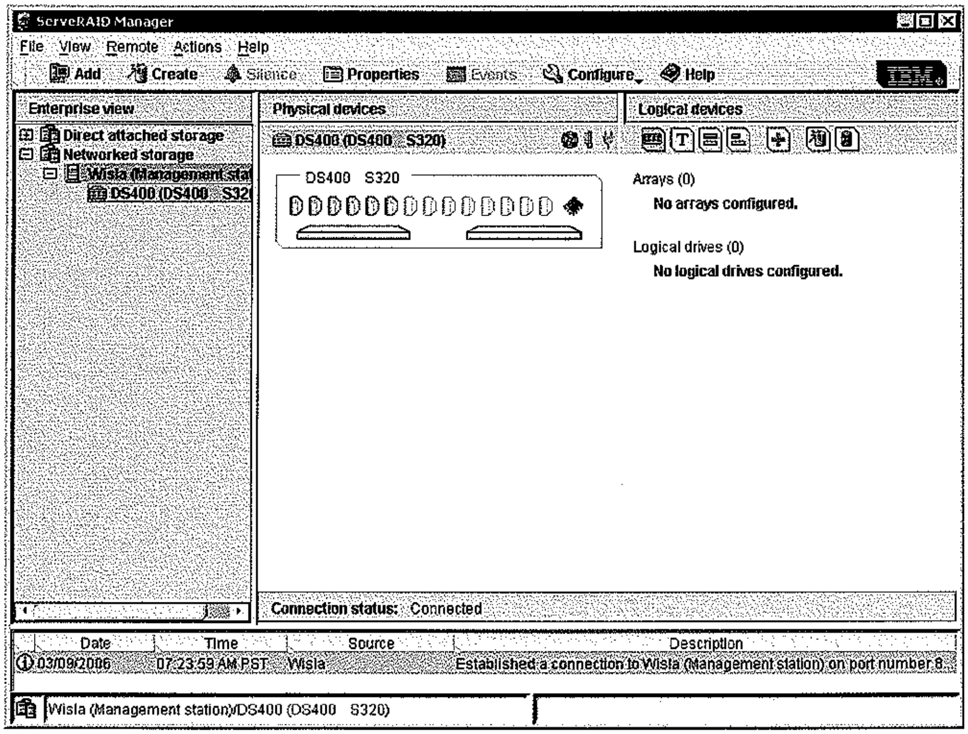


Figure 22 DS400 storage as seen via ServeRAID Manager

0.0.1 Update DS400 controller firmware

Complete the following steps to update the firmware image on the DS400 controller using the ServeRAID Manager program:

1. In the Enterprise view of the ServeRAID Manager program interface expand the ServeRAID agent.

2. Right-click **Networked storage**, then select **Update controller images**, as shown in Figure 23. This opens the Software Update wizard.

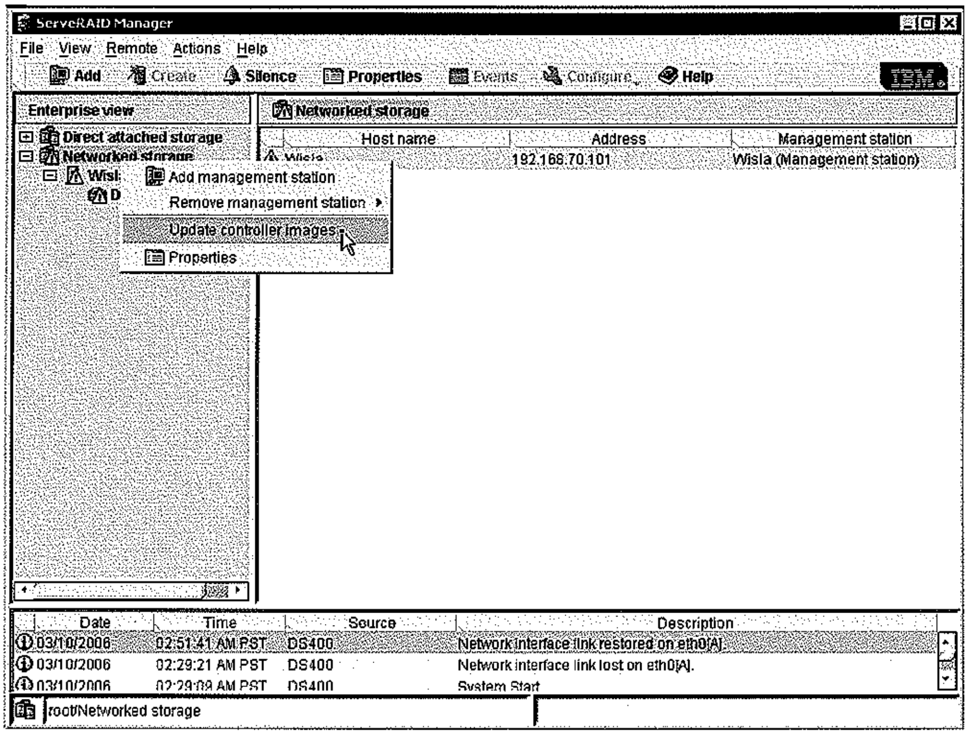


Figure 23 Update controller image

3. When prompted, browse to the folder containing the firmware that was downloaded and unzipped earlier in “Latest DS400 firmware” on page 8.
4. Choose the file to be added, as shown in Figure 24 on page 24.

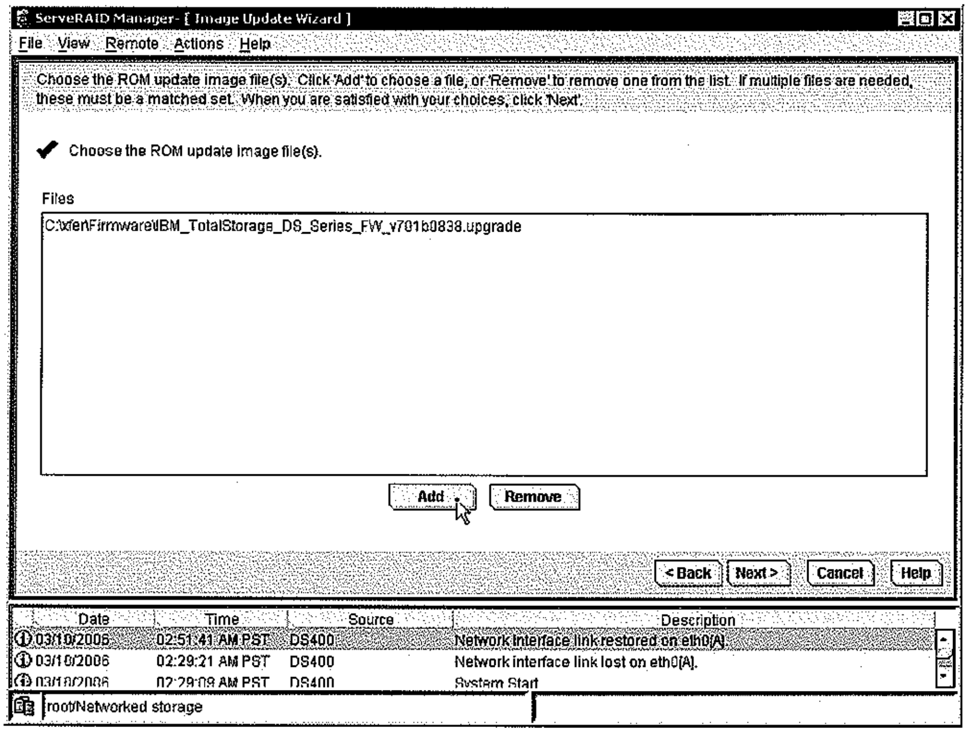


Figure 24 Add firmware file

5. Choose the controller to update and click **Next**, as shown in Figure 25.

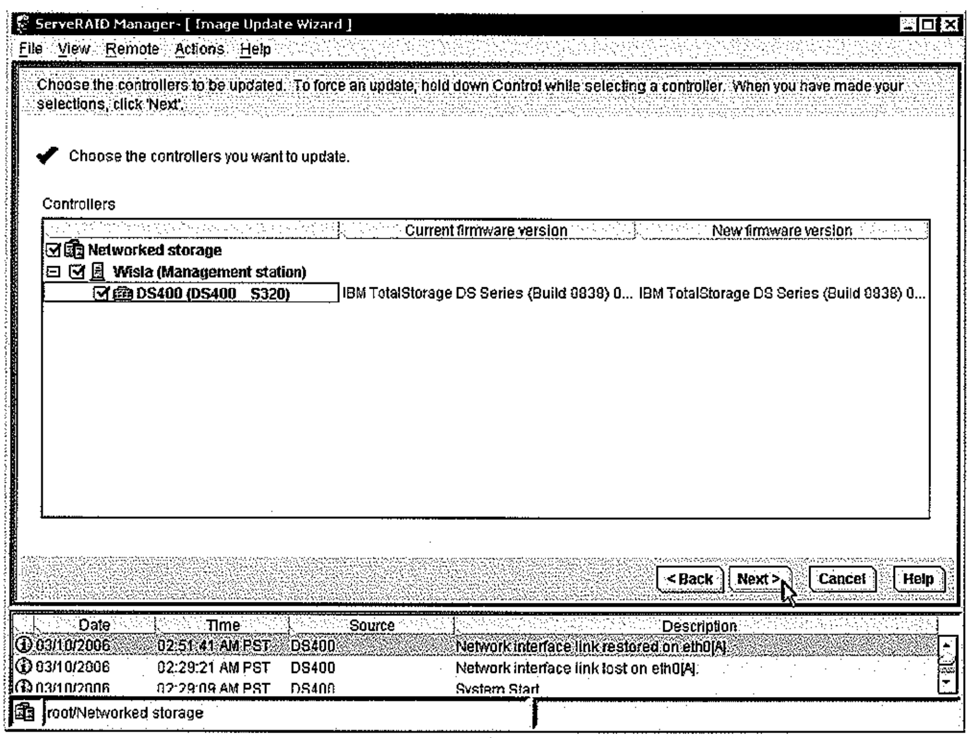


Figure 25 Choose controller to update

6. In the next screen click **Apply**, as in Figure 26.

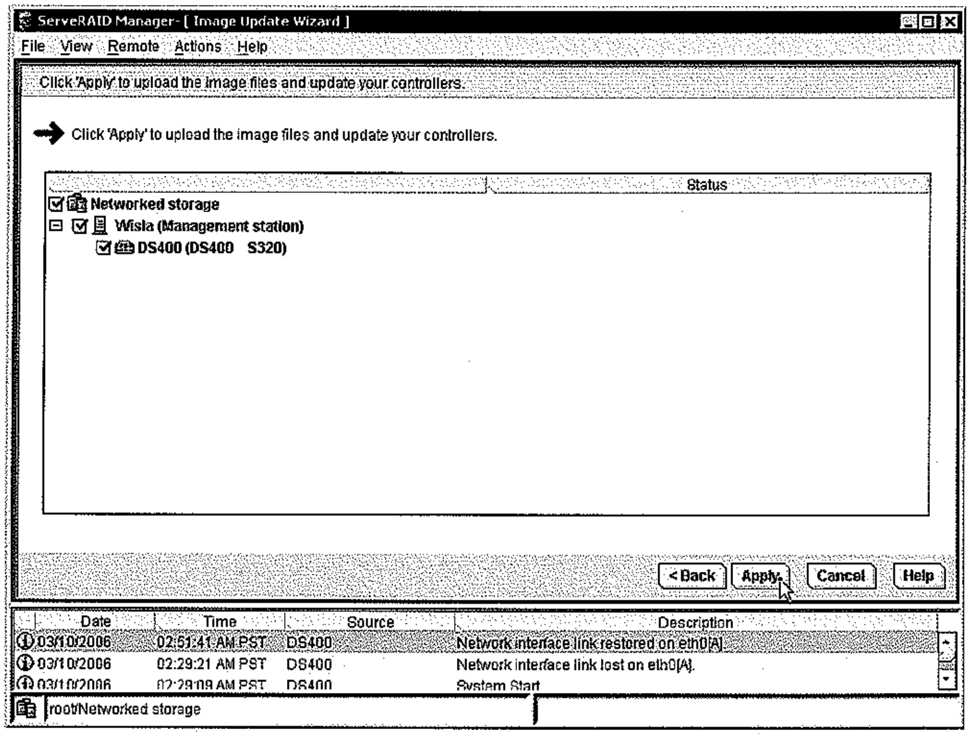


Figure 26 Apply settings

7. Verify the ROM update, as in Figure 27.

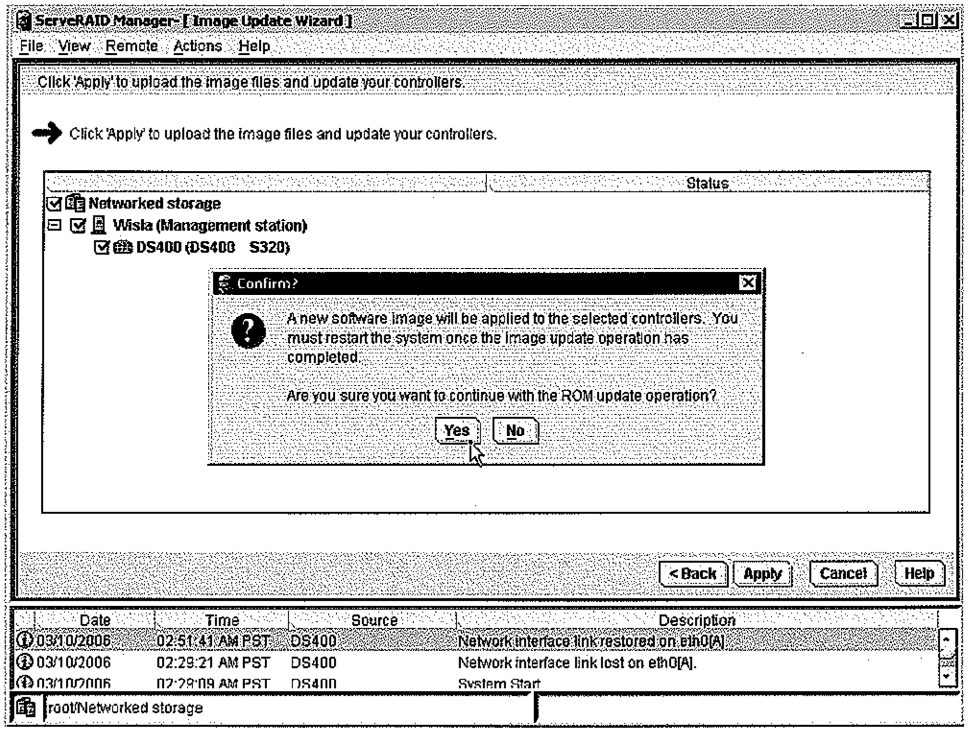


Figure 27 Verify ROM update

8. Once the update is complete, restart the enclosure by right-clicking the controller in the Enterprise view of the ServeRAID Manager program interface, as shown in Figure 28.

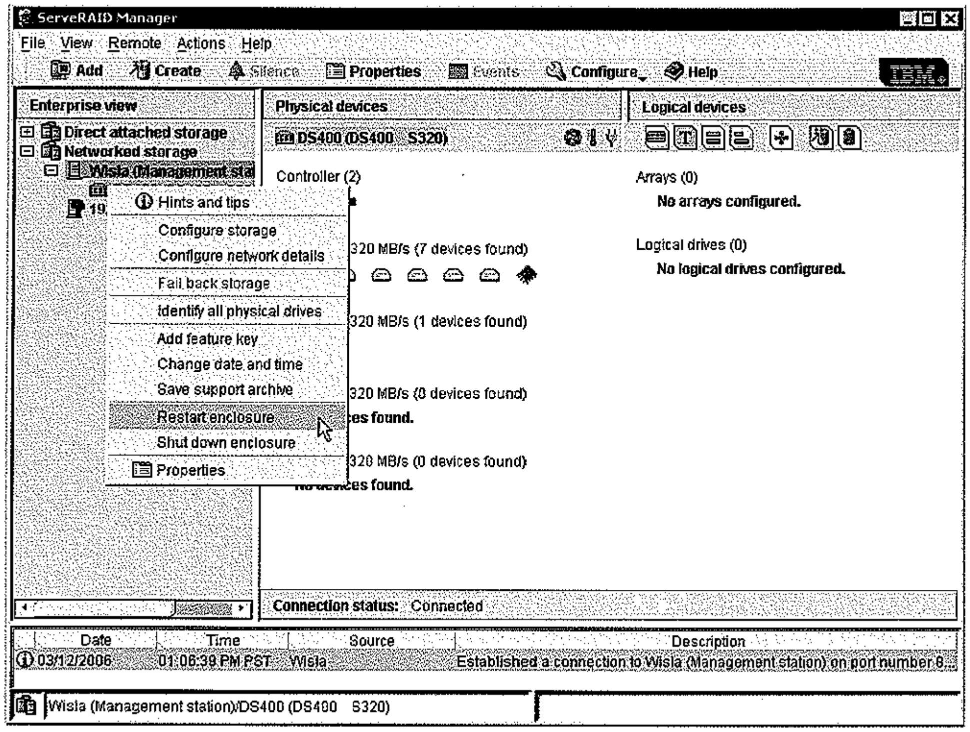


Figure 28 Finish and restart controller

Once the controller reboots, the SAN is installed and configured.

SUSMAN GODFREY L.L.P.
A REGISTERED LIMITED LIABILITY PARTNERSHIP
ATTORNEYS AT LAW
5TH FLOOR
654 MADISON AVENUE
NEW YORK, NEW YORK 10065
WWW.SUSMANGODFREY.COM

SUITE 5100
901 MAIN STREET
DALLAS, TEXAS 75202-3775
(214) 754-1900

SUITE 950
1901 AVENUE OF THE STARS
LOS ANGELES, CALIFORNIA 90067-6029
(310) 789-3100

SUITE 3800
1201 THIRD AVENUE
SEATTLE, WASHINGTON 98101-3000
(206) 516-3880

SUITE 5100
1000 LOUISIANA STREET
HOUSTON, TEXAS 77002-5096
(713) 651-9366

TIBOR L. NAGY JR.
DIRECT DIAL (212) 336-8332

DIRECT DIAL FAX (212) 336-8340
E-MAIL TNAGY@SUSMANGODFREY.COM

By Hand Delivery

May 2, 2008

United States Courthouse
Chambers of the Hon. Lewis A. Kaplan
500 Pearl Street
New York, NY 10007

Re: *IBM v. PSI*, 06-cv-13565 / Dr. Patt & Dr. Patel

Dear Judge Kaplan:

In light of IBM's letter yesterday some brief background on Dr. Yale Patt, one of PSI's consulting experts in this matter, is in order. Dr. Patt is a world-renowned computer architect and has been a professor for over 40 years. On April 18, 2008, counsel for IBM asked ***Dr. Patt*** whether ***he*** would like to serve as technical advisor to the Court in this action. Dr. Patt declined, in light of his previously having been retained by PSI.

Dr. Patt and Dr. Patel have the following relationship: (1) from 1995-1999, Dr. Patt served as Dr. Patel's Ph.D advisor (for perspective, Dr. Patt has had 28 other Ph.D advisees since 1995); and (2) Dr. Patt and Dr. Patel have co-authored several publications, including a textbook on computer architecture. These publications are listed on the CV of Dr. Patel that IBM sent to PSI on April 28, 2008.

IBM has now withdrawn its support of Dr. Patel because IBM says it needs "more information from PSI's counsel regarding Dr. Patt's role in the case." This position is unreasonable. If Dr. Patel had been appointed technical advisor at the outset of this case, that decision would not have restricted either party's right to hire Dr. Patt (or any other former professor of Dr. Patel) as an expert in this litigation. Dr. Patt may or may not write an expert report in this case, and he may or may not testify—those decisions will depend on, among other things, the roles that IBM's experts will play. In any case, the precise role that Dr. Patt plays is irrelevant: objecting to Dr. Patel because he studied under Dr. Patt 10 years ago is not a good-faith objection.

IBM has now expanded the universe of technical-advisor conflicts to include any prior professional relationship with Hewlett-Packard Corporation, Microsoft Corporation, Intel Corporation, or Professor Yale Patt. Contrary to its letter of yesterday, IBM is not trying to work with PSI to find a mutually acceptable candidate—it is, instead, making unreasonable objections to any candidate PSI supports, even those IBM itself proposed. The Court should select a technical advisor from among Dr. Fisher, Dr. Wakerly and Dr. Patel. And PSI should not have to expend further resources searching for and evaluating candidates that IBM is not willing to reasonably accept.

Sincerely,

SUSMAN GODFREY LLP

Tibor L. Nagy, Jr. (TN-9569)
tnagy@susmangodfrey.com
Attorney for PSI

cc: Edward Defranco, *Counsel for IBM*

What is interrupt processing?

An *interrupt* is an event that alters the sequence in which the processor executes instructions.

An interrupt might be planned (specifically requested by the currently running program) or unplanned (caused by an event that might or might not be related to the currently running program). z/OS® uses six types of interrupts, as follows:

Supervisor calls or SVC interrupts
: These interrupts occur when the program issues an SVC to request a particular system service. An SVC interrupts the program being executed and passes control to the supervisor so that it can perform the service. Programs request these services through macros such as OPEN (open a file), GETMAIN (obtain storage), or WTO (write a message to the system operator).

I/O interrupts
: These interrupts occur when the channel subsystem signals a change of status, such as an input/output (I/O) operation completing, an error occurring, or an I/O device such as a printer has become ready for work.

External interrupts
: These interrupts can indicate any of several events, such as a time interval expiring, the operator pressing the interrupt key on the console, or the processor receiving a signal from another processor.

Restart interrupts
: These interrupts occur when the operator selects the restart function at the console or when a restart SIGP (signal processor) instruction is received from another processor.

Program interrupts
: These interrupts are caused by program errors (for example, the program attempts to perform an invalid operation), **page faults** (the program references a page that is not in central storage), or requests to monitor an event.

Machine check interrupts
: These interrupts are caused by machine malfunctions.

When an interrupt occurs, the hardware saves pertinent information about the program that was interrupted and, if possible, disables the processor for further interrupts of the same type. The hardware then routes control to the appropriate interrupt handler routine. The program status word or PSW is a key resource in this process.

The program status word (PSW) is a 128-bit data area in the processor that, along with a variety of other types of registers (control registers, timing registers, and prefix registers) provides details crucial to both the hardware and the software. The current PSW includes the address of the next program instruction and control information about the program that is running. Each processor has only one current PSW. Thus, only one task can execute on a processor at a time.

The PSW controls the order in which instructions are fed to the processor, and indicates the status of the system in relation to the currently running program. Although each processor has only one PSW, it is useful to think of three types of PSWs to understand interrupt processing:

- Current PSW
- New PSW

- Old PSW

The current PSW indicates the next instruction to be executed. It also indicates whether the processor is enabled or disabled for I/O interrupts, external interrupts, machine check interrupts, and certain program interrupts. When the processor is enabled, these interrupts can occur. When the processor is disabled, these interrupts are ignored or remain pending.

There is a new PSW and an old PSW associated with each of the six types of interrupts. The new PSW contains the address of the routine that can process its associated interrupt. If the processor is enabled for interrupts when an interrupt occurs, PSWs are switched using the following technique:

1. Storing the current PSW in the old PSW associated with the type of interrupt that occurred.
2. Loading the contents of the new PSW for the type of interrupt that occurred into the current PSW.

The current PSW, which indicates the next instruction to be executed, now contains the address of the appropriate routine to handle the interrupt. This switch has the effect of transferring control to the appropriate interrupt handling routine.

Mainframe architecture provides registers to keep track of things. The PSW, for example, is a register used to contain information that is required for the execution of the currently active program.

Mainframes provide other registers, as follows:

Access registers
: These registers specify the address space in which data is found.

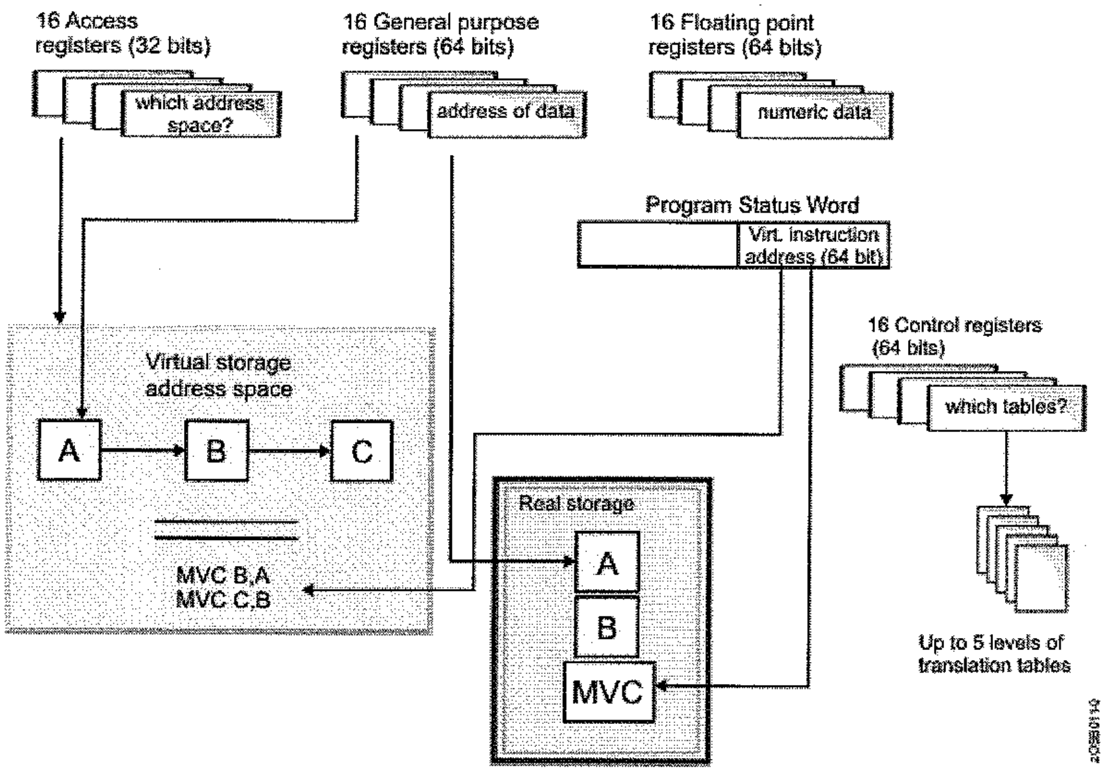
General registers
: These registers address data in storage, and also hold user data.

Floating point registers
: These registers hold numeric data in floating point form.

Control registers
: These registers are used by the operating system itself, for example, as references to translation tables.

Figure 1 shows the use of registers and the PSW.

Figure 1. Registers and the PSW



Parent topic: Supervising the execution of work in the system
Next topic: Creating dispatchable units of work

Notices | Terms of use | Support | Contact z/OS

URL: http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.zconcepts.doc/zconc_interrupts.html

IBM.

IBM Terminology

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #

Please send any feedback about the terms and definitions on this site to terms@ca.ibm.com

E

E1
: A digital trunking facility standard used in Europe and elsewhere, capable of transmitting and receiving 30 digitized voice or data channels. Two additional channels are used for synchronization, framing, and signaling. The transmission rate is 2048 kilobits per second. See also T1.

EAB
: See Enterprise Address Book.

EAC
: See estimate at completion.

EA-enabled table space
: A table space or index space that is enabled for extended addressability and that contains individual partitions (or pieces, for LOB table spaces) that are greater than 4 GB.

E and M
: A channel associated signaling protocol in which signaling is done using two leads: an M-lead that transmits battery or ground and an E-lead that receives open or ground.

EAO exception
: See effective address overflow exception.

EAR
: See enterprise archive.

EAR file
: See enterprise archive.

early resource release
: The release of resources (such as devices, volumes, and data sets) after they are no longer needed.

early token release
: A function, supported by token-ring adapter types 2 and 3, that allows a transmitting station to release the token after transmitting the ending delimiter.

earned value
: A measure of the value of work performed so far. Earned value uses original estimates and progress-to-date to show whether the actual costs incurred are on budget and whether the tasks are ahead or behind the baseline plan.

EAR project
: See enterprise application project.

eavesdropping
: A breach of communication security in which the information remains intact, but its privacy is compromised. See also impersonation, tampering.

EB

occupied by or reserved for a particular data set, data space, or file.

FlashCopy relationship
: See FlashCopy mapping.

FlashCopy service
: A copy service that duplicates the contents of a source virtual disk (VDisk) on a target VDisk. In the process, the original contents of the target VDisk are lost. See also point-in-time copy.

flash memory
: A computer chip with a read-only memory that retains its data when the power is turned off and that can be electronically erased and reprogrammed without being removed from the circuit board.

flat browse
: A browse of the descendant activities of a specified process, on which each descendant activity can be returned exactly once.

flat business object
: A business object that contains only simple attributes and does not contain any child business objects. See also hierarchical business object.

flat collection
: A collection that has no hierarchical structure.

flatten
: (1) To display multiple iterations of data in a single window.
: (2) See demarshal.

flight recorder
: An object that stores trace information used to record a history of what has happened in the system's programs. The flight recorder contains only information that helps to identify the flow of the system's programs and status information.

float constant
: (1) A number containing a decimal point, an exponent, or both a decimal point and an exponent. The exponent contains an "e" or "E," an optional sign (+ or -), and one or more digits (0 through 9).
: (2) A constant representing a non-integral number.

floating bar chart
: In the GDDM function, a chart that shows bars detached from either line. See also composite bar chart, multiple bar chart.

floating bar graph
: In Performance Tools, a graph that shows bars detached from either line. See also composite bar graph.

floating currency symbol
: A currency symbol that appears immediately to the left of the far left position in an edited field. See also fixed currency symbol.

floating overlay
: See page overlay.

floating point
: A method of encoding binary floating-point numbers and decimal floating-point numbers within the limits of finite precision available on computers.